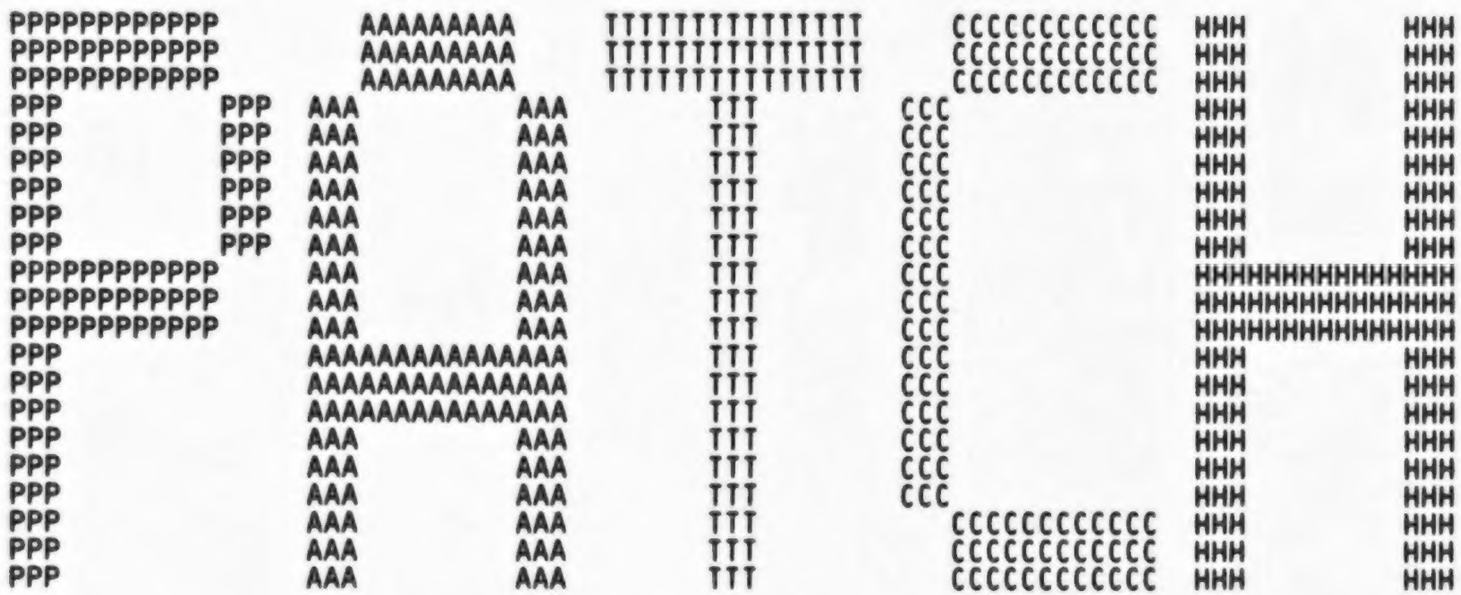
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH

```
PPPPPPPP      AAAAAA    TTTTTTTTTT  EEEEEEEEEE  XX      XX    AAAAAA
PPPPPPPP      AAAAAA    TTTTTTTTTT  EEEEEEEEEE  XX      XX    AAAAAA
PP      PP   AA    AA       TT      EE          XX      XX   AA    AA
PP      PP   AA    AA       TT      EE          XX      XX   AA    AA
PP      PP   AA    AA       TT      EE            XX  XX     AA    AA
PP      PP   AA    AA       TT      EE            XX  XX     AA    AA
PPPPPPPP     AA    AA       TT      EEEEEEEE        XX       AA    AA
PPPPPPPP     AA    AA       TT      EEEEEEEE        XX       AA    AA
PP           AAAAAAAAAA     TT      EE            XX  XX     AAAAAAAAAA
PP           AAAAAAAAAA     TT      EE            XX  XX     AAAAAAAAAA
PP           AA    AA       TT      EE          XX      XX   AA    AA      ....
PP           AA    AA       TT      EE          XX      XX   AA    AA      ....
PP           AA    AA       TT      EEEEEEEEEE  XX      XX   AA    AA      ....
PP           AA    AA       TT      EEEEEEEEEE  XX      XX   AA    AA      ....
```

```
LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE PATEXA (
;     2        L 0002  0                 %IF %VARIANT EQL 1
;     3          0003  0                 %THEN
;     4          0004  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
;     5          0005  0                 %FI
;     6          0006  0                 IDENT = 'V04-000') =
;     7          0007  1 BEGIN
;     8          0008  1
;     9          0009  1 !*****************************************************************************
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    12          0012  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    13          0013  1 !*  ALL RIGHTS RESERVED.                                                      *
;    14          0014  1 !*                                                                           *
;    15          0015  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    16          0016  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    17          0017  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    18          0018  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    19          0019  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    20          0020  1 !*  TRANSFERRED.                                                              *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    23          0023  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    24          0024  1 !*  CORPORATION.                                                              *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    27          0027  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*****************************************************************************
;    31          0031  1 !
;    32          0032  1 ! FACILITY:     PATCH
;    33          0033  1 !
;    34          0034  1 !++
;    35          0035  1 ! FUNCTIONAL DESCRIPTION:
;    36          0036  1 !
;    37          0037  1 !       EXAMINE, DEPOSIT, AND DELETE ROUTINES FOR STARLET PATCH FACILITY
;    38          0038  1 !
;    39          0039  1 ! History:
;    40          0040  1 !       Author: Carol Peters, 21 Jul 1976: Version 01
;    41          0041  1 !
;    42          0042  1 !               Kathleen Morse, 19 Oct 1977: Version X01.00
;    43          0043  1 !
;    44          0044  1 ! Modified by:
;    45          0045  1 !
;    46          0046  1 !       V03-002 MTR0016         Mike Rhodes             02-Nov-1982
;    47          0047  1 !               Modify routine RELOCAT_INS to pass the address of the
;    48          0048  1 !               the instruction(s) to be relocated to the patch area.
;    49          0049  1 !               This address will be passed initially to PAT$EXP_AREA
;    50          0050  1 !               which in turn may call routine PAT$BUILD_ISE (which is
;    51          0051  1 !               called to create the default patch area if one does not
;    52          0052  1 !               already exist).  PAT$BUILD_ISE will use this address to
;    53          0053  1 !               propagate the image section attributes of the patched
;    54          0054  1 !               image section to the newly created default patch area.
;    55          0055  1 !
;    56          0056  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
;    57          0057  1 !               Modify file names to remove duplicate file name useage
```

```
;    58          0058  1 !                 between code and require files.
;    59          0059  1 !
;    60          0060  1 !           V02-023 PCG0001         Peter George            04-FEB-1981
;    61          0061  1 !                   Add require statement for LIB$:PATDEF.REQ
;    62          0062  1 !
;    63          0063  1 !           V0122   BLS0039         Benn Schreiber          3-Feb-1981
;    64          0064  1 !                   Correct handling of patch area.
;    65          0065  1 !
;    66          0066  1 !           V0121   CNH0014         Chris Hume              21-Sep-1979     11:00
;    67          0067  1 !                   Added relocation support for the ACBG and ACBH instructions.
;    68          0068  1 !
;    69          0069  1 !           V0120   CNH0008         Chris Hume              28-Jun-1979     14:00
;    70          0070  1 !                   Fix CASE replacement bug and disallow relocation of these
;    71          0071  1 !                   instructions.  (PATMAI.B32 V0222, PATACT.B32 V0124,
;    72          0072  1 !                   PATMAC.B32 V0217, PATMSG.MDL V0202)
;    73          0073  1 !
;    74          0074  1 ! Revision history:
;    75          0075  1 !  NO   DATE            PROGRAMMER              PURPOSE
;    76          0076  1 !  --   ----            ----------              -------
;    77          0077  1 !
;    78          0078  1 !  00   19-OCT-77       K.D. MORSE              ADAPT VERSION 49 FOR PATCH
;    79          0079  1 !  01   01-DEC-77       K.D. MORSE              ADD DELETE ROUTINE.
;    80          0080  1 !  02   27-DEC-77       K.D. MORSE              CHANGE PAT$OUT_VALUE CALLS. (57)
;    81          0081  1 !                                              ADD PAT$SYM_DEPOS. (58)
;    82          0082  1 !  03   2-JAN-78        K.D. MORSE              ADD CHECK FOR NO SYMBOLS IN IMAGE.
;    83          0083  1 !  04   3-JAN-78        K.D. MORSE              ADD CHECK FOR NO PATCHAREA
;    84          0084  1 !                                              ALLOCATED BEFORE DEPOSIT /PAT.
;    85          0085  1 !  05   4-JAN-78        K.D. MORSE              NO CHANGES FOR VERS 50-53.
;    86          0086  1 !  06   5-JAN-78        K.D. MORSE              CHANGE PAT$INS_DECODE_CALLS. (54)
;    87          0087  1 !                                              NO CHANGES FOR VERS 55,56.
;    88          0088  1 !  07   24-JAN-78       K.D. MORSE              NO CHANGES FOR VERS 59.
;    89          0089  1 !  08   27-JAN-78       K.D. MORSE              ADD CHECK FOR EXIT_TOKEN IN
;    90          0090  1 !                                              PAT$REPLACE_CMD TO RECOGNIZE
;    91          0091  1 !                                              END OF OLD LIST.
;    92          0092  1 !  09   28-JAN-78       K.D. MORSE              BUILD REPLACEMENT CODE INTO
;    93          0093  1 !                                              TEMPORARY BUFFER.
;    94          0094  1 !  10   01-MAR-78       K.D. MORSE              CHANGE ERRONEOUS PAT$_DECODE
;    95          0095  1 !                                              ERROR MSGS TO PAT$_ENCODE.
;    96          0096  1 !  11   24-MAR-78       K.D. MORSE              NONE FOR VERS 60-61.
;    97          0097  1 !  12   04-APR-78       K.D. MORSE              NONE FOR VERS 62.
;    98          0098  1 !  13   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
;    99          0099  1 !  14   28-APR-78       K.D. MORSE              ADD ASSEMBLER DIRECTIVE FLAG
;   100          0100  1 !                                              TO PAT$OUT_MEM_LOC.
;   101          0101  1 !  15   18-MAY-78       K.D. MORSE              NO CHANGES FOR VERS 63.
;   102          0102  1 !  16   26-MAY-78       K.D. MORSE              ADD CODE TO ALLOW FORWARD
;   103          0103  1 !                                              REFERENCING IN SYMBOLIC
;   104          0104  1 !                                              INSTRUCTION OPERANDS.
;   105          0105  1 !  17   13-JUN-78       K.D. MORSE              ADD FAO COUNTS TO SIGNALS.
;   106          0106  1 !  18   19-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 64.
;   107          0107  1 !  19   28-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 65-67.
;   108          0108  1 !                                              ADD CODE FOR EV/LITERAL AND
;   109          0109  1 !                                              ROUTINE DISPLAY LVTS. (66)
;   110          0110  1 !                                              NO CHANGES FOR VERS 69-74.
;   111          0111  1 !
;   112          0112  1 !--
```

```
;  114     0113  1 FORWARD ROUTINE
;  115     0114  1           PAT$DEPOSIT_CMD : NOVALUE,                    ! Deposits a datum into an address
;  116     0115  1           PAT$EXAMINE_CMD : NOVALUE,                    ! Examines a location
;  117     0116  1           PAT$REPLACE_CMD : NOVALUE,                    ! Replaces an instruction
;  118     0117  1           RELOCAT_INS : NOVALUE,                        ! Relocates instructions to patch area
;  119     0118  1           PAT$SUBST_INS,                                ! Substitutes instructions in patch area
;  120     0119  1           PAT$OUT_MEM_LOC,                              ! Outputs the contents of a memory location
;  121     0120  1           DISPLAY_LVTS : NOVALUE,                       ! Search LVT and display pathnames
;  122     0121  1           PAT$REG_MATCH,                                ! Matches a string to a register name
;  123     0122  1           PAT$FILL_BUF : NOVALUE;                       ! Writes data into temporary buffers
;  124     0123  1
;  125     0124  1 LIBRARY 'SYS$LIBRARY:LIB.L32';                          ! System definitions
;  126     0125  1 REQUIRE 'SRC$:VXSMAC.REQ';
;  127     0190  1 REQUIRE 'SRC$:BSTRUC.REQ';
;  128     0266  1 REQUIRE 'SRC$:LISTEL.REQ';
;  129     0308  1 REQUIRE 'SRC$:PATPCT.REQ';
;  130     0348  1 REQUIRE 'SRC$:PATGEN.REQ';
;  131     0570  1 REQUIRE 'LIB$:PATDEF.REQ';                              ! Defines literals
;  132     0624  1 REQUIRE 'LIB$:PATMSG.REQ';
;  133     0798  1 REQUIRE 'SRC$:SYSLIT.REQ';
;  134     0848  1 REQUIRE 'SRC$:PATRTS.REQ';
;  135     1944  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R1976  1     SWITCHES LIST (SOURCE);
;  R1977  1
;  R1978  1     EXTERNAL ROUTINE
;  R1979  1              PAT$fao_out;            ! formats a line and outputs to the terminal
;  R1980  1
```

```
;  136     2026  1 REQUIRE 'SRC$:PATTER.REQ';
;  137     2233  1 REQUIRE 'SRC$:PREFIX.REQ';
;  138     2421  1 REQUIRE 'SRC$:PATPRE.REQ';
;  139     2584  1 REQUIRE 'SRC$:VAXOPS.REQ';
;  140     2798  1
;  141     2799  1 EXTERNAL
;  142     2800  1         PAT$GB_SYMBOLS,                 ! Indicator if image had symbols
;  143     2801  1         PAT$GL_OLD_ASD,                 ! Descriptor for old contents assembler dire
;  144     2802  1         PAT$GL_NEW_ASD,                 ! Descriptor for new contents assembler dire
;  145     2803  1         PAT$GL_TEMP_BUF : BLOCK[,BYTE], ! Descriptor for temporary buffer for deposi
;  146     2804  1         PAT$GL_RLOC_BUF : BLOCK[,BYTE], ! Descriptor for relocated instruction strea
;  147     2805  1         PAT$GB_SUBST_IN : VECTOR[,BYTE], ! Buffer for substitution instruction stream
;  148     2806  1         PAT$GL_BR_DISPL,                ! Branch displacement that does not fit
;  149     2807  1         PAT$GL_PATAREA : REF BLOCK[,BYTE], ! Pointer to patch area descriptor
;  150     2808  1         PAT$GL_IMGHDR : REF BLOCK[,BYTE], ! Pointer to image header
;  151     2809  1         PAT$GB_LOC_TYPE: BYTE,          ! Type of end range argument
;  152     2810  1         PAT$GB_MOD_PTR: REF VECTOR [, BYTE], ! Pointer to mode level
;  153     2811  1         PAT$GL_IHPPTR : REF BLOCK[,BYTE], ! Pointer to image header patch area
;  154     2812  1         PAT$CP_OUT_STR : REF VECTOR[,BYTE], ! Points into current output buffer
;  155     2813  1         PAT$GL_CONTEXT: BITVECTOR,      ! Context bits longword
;  156     2814  1         PAT$GL_BUF_SIZ,                 ! Holds count in output buffer
;  157     2815  1         PAT$GL_HEAD_LST,                ! Head of linked list of expressions
;  158     2816  1         PAT$GL_LAST_LOC,                ! Last location displayed
;  159     2817  1         PAT$GL_LAST_VAL,                ! Last value displayed
;  160     2818  1         PAT$GL_NEXT_LOC,                ! Next location to display
;  161     2819  1         PAT$GL_SYMTBPTR,                ! Pointer to current symbol table
;  162     2820  1         PAT$GL_OLDLABLS,                ! Listhead for old contents labels (from cur
;  163     2821  1         PAT$GL_NEWLABLS,                ! Listhead for new contents un-relocated lab
;  164     2822  1         PAT$GL_RLCLABLS;                ! List head for new contents relocated label
;  165     2823  1
;  166     2824  1 EXTERNAL ROUTINE
;  167     2825  1         PAT$ADD_LABELS : NOVALUE,       ! Adds labels to user-defined symbol table
;  168     2826  1         PAT$ADD_NT_T_PV : NOVALUE,      ! Build pathname vectors from NT_PTRs
;  169     2827  1         PAT$EXP_AREA : NOVALUE,         ! Expands patch area
;  170     2828  1         PAT$FAO_PUT : NOVALUE,          ! Formats buffered output
;  171     2829  1         PAT$FREERELEASE,                ! Deallocates free memory
;  172     2830  1         PAT$FREEZ,                      ! Allocates and zeroes free memory
;  173     2831  1         PAT$GET_NXT_LVT,                ! Provide access to the LVT
;  174     2832  1         PAT$GET_VALUE : NOVALUE,        ! Gets byte stream of values from image
;  175     2833  1         PAT$INS_DECODE,                 ! Routine to output memory as
;  176     2834  1                                         ! symbolic instructions
;  177     2835  1         PAT$INS_ENCODE,                 ! Routine to encode a symbolic instruction
;  178     2836  1         PAT$MAP_ADDR : NOVALUE,         ! Computes mapped addresses
;  179     2837  1         PAT$OUT_NUM_VAL,                ! Outputs numeric values
;  180     2838  1         PAT$OUT_PUT : NOVALUE,          ! Actually does the terminal I/O
;  181     2839  1         PAT$OUT_SYM_VAL,                ! Outputs symbol name with value
;  182     2840  1         PAT$PRINT_PATH : NOVALUE,       ! Print out pathnames
;  183     2841  1         PAT$RESOLVE_INS : NOVALUE,      ! Resolves forward references in symbolic in
;  184     2842  1         PAT$SYMBOL_VALU,                ! Finds the value bound to a symbol name
;  185     2843  1         PAT$UNMAP_ADDR : NOVALUE,       ! Computes unmapped addresses
;  186     2844  1         PAT$VAL_TO_SYM,                 ! Translates a value to a symbol name
;  187     2845  1         PAT$WRITE_MEM;                  ! Routine to write to user's memory
```

```
;  189          2846  1 !++
;  190          2847  1 !
;  191          2848  1 ! REGISTER_TABLE holds one entry per register.  Each entry is made
;  192          2849  1 ! up of one longword.  The first byte holds the character count of
;  193          2850  1 ! the register name.  The second through fourth bytes hold the
;  194          2851  1 ! register name string.  A sample entry follows:
;  195          2852  1 !
;  196          2853  1 !       +---------------------------------------------------------------+
;  197          2854  1 !       !               !       0       !       R       !       2       !
;  198          2855  1 !       +---------------------------------------------------------------+
;  199          2856  1 !
;  200          2857  1 !--
;  201          2858  1
;  202          2859  1 MACRO
;  203        M 2860  1         REGISTER_ENTRY (STRING) =
;  204          2861  1                 %CHARCOUNT (STRING), %ASCII STRING, REP 3 - %CHARCOUNT (STRING) OF BYTE (0)%;
;  205          2862  1
;  206          2863  1 BIND
;  207          2864  1         REGISTER_TABLE = UPLIT BYTE (
;  208          2865  1                 REGISTER_ENTRY ('R0'),
;  209          2866  1                 REGISTER_ENTRY ('R1'),
;  210          2867  1                 REGISTER_ENTRY ('R2'),
;  211          2868  1                 REGISTER_ENTRY ('R3'),
;  212          2869  1                 REGISTER_ENTRY ('R4'),
;  213          2870  1                 REGISTER_ENTRY ('R5'),
;  214          2871  1                 REGISTER_ENTRY ('R6'),
;  215          2872  1                 REGISTER_ENTRY ('R7'),
;  216          2873  1                 REGISTER_ENTRY ('R8'),
;  217          2874  1                 REGISTER_ENTRY ('R9'),
;  218          2875  1                 REGISTER_ENTRY ('R10'),
;  219          2876  1                 REGISTER_ENTRY ('R11'),
;  220          2877  1                 REGISTER_ENTRY ('AP'),
;  221          2878  1                 REGISTER_ENTRY ('FP'),
;  222          2879  1                 REGISTER_ENTRY ('SP'),
;  223          2880  1                 REGISTER_ENTRY ('PC'),
;  224          2881  1                 REGISTER_ENTRY ('PSL')):
;  225          2882  1
;  226          2883  1         BLOCK [, LONG];
;  227          2884  1
;  228          2885  1 !++
;  229          2886  1 ! These field definitions control access to the register table.
;  230          2887  1 !--
;  231          2888  1 MACRO
;  232          2889  1         REG_NAME        =8, 24, 0%,
;  233          2890  1         CTD_REG_NAME    =0, 24, 0%,
;  234          2891  1         REG_CH_CNT      =0,  8, 0%;
;  235          2892  1
;  236          2893  1 !++
;  237          2894  1 ! Common ascii counted strings used in FAO calls.
;  238          2895  1 !--
;  239          2896  1
;  240          2897  1 BIND
;  241          2898  1         CS_ASCII        = UPLIT ( %ASCIC '''!AD'''),
;  242          2899  1         COLON_TAB_STG   = UPLIT ( %ASCIC ':  '),
;  243          2900  1         CAR_CTL_STG     = UPLIT ( %ASCIC '!/'),
;  244          2901  1         OLD_TAB_STG     = UPLIT ( %ASCIC 'old:  '),
;  245          2902  1         NEW_TAB_STG     = UPLIT ( %ASCIC 'new:  ');
```

```
;  246          2903  1
;  247          2904  1 LITERAL
;  248          2905  1         NO_CASE_TABLE   = 0,                    ! Don't print case dispatch table
;  249          2906  1         CASE_TABLE      = 1;                    ! Print CASE dispatch table
```

```
;  251     2907  1 SWITCHES NOOPTIMIZE;
;  252     2908  1 GLOBAL ROUTINE PAT$DEPOSIT_CMD : NOVALUE =
;  253     2909  1
;  254     2910  1 !++
;  255     2911  1 ! FUNCTIONAL DESCRIPTION:
;  256     2912  1 !
;  257     2913  1 !       This routine handles all delete and deposit commands, those for
;  258     2914  1 !       instructions and those for values.  The instruction(s)/value(s)
;  259     2915  1 !       specified in the delete command must be identical to those actually
;  260     2916  1 !       contained in the location(s), otherwise an error message is produced and
;  261     2917  1 !       the command ends prematurely.  For a deposit command, the instruction(s)
;  262     2918  1 !       /value(s) specified are written to consecutive locations.  The image is
;  263     2919  1 !       not modified in either case unless the entire command works.
;  264     2920  1 !
;  265     2921  1 !       The command argument list is made up of entries that are
;  266     2922  1 !       each three longwords long. The first is a forward link to the
;  267     2923  1 !       next entry. The second longword in the first entry in the list
;  268     2924  1 !       is the address into which some value(s) is (are) to be deleted or deposited.
;  269     2925  1 !       The third longword is unused. The second longword in the second
;  270     2926  1 !       and subsequent entries are the values to be deleted or deposited.
;  271     2927  1 !       The first value should be the contents of the specified location;
;  272     2928  1 !       the second, the contents of that location plus the current mode_length, etc.
;  273     2929  1 !
;  274     2930  1 !       For instructions, the increment is the length of each subsequent
;  275     2931  1 !       instruction. The second longword contains the address of a counted byte
;  276     2932  1 !       stream that is to be translated into a binary instruction which is the
;  277     2933  1 !       old contents of that location.
;  278     2934  1 !
;  279     2935  1 ! CALLING SEQUENCE:
;  280     2936  1 !
;  281     2937  1 !       PAT$DEPOSIT_CMD ()
;  282     2938  1 !
;  283     2939  1 ! INPUTS:
;  284     2940  1 !
;  285     2941  1 !       none
;  286     2942  1 !
;  287     2943  1 ! IMPLICIT INPUTS:
;  288     2944  1 !
;  289     2945  1 !       PAT$GL_HEAD_LST, the head of the linked PATCH command argument list.
;  290     2946  1 !       The current mode.
;  291     2947  1 !
;  292     2948  1 ! OUTPUTS:
;  293     2949  1 !
;  294     2950  1 !       none
;  295     2951  1 !
;  296     2952  1 ! IMPLICIT OUTPUTS:
;  297     2953  1 !
;  298     2954  1 !       The values PAT$GL_LAST_LOC, PAT$GL_LAST_VAL, and PAT$GL_NEXT_LOC
;  299     2955  1 !       are set after each deposit is done.
;  300     2956  1 !
;  301     2957  1 ! ROUTINE VALUE:
;  302     2958  1 !
;  303     2959  1 !       novalue
;  304     2960  1 !
;  305     2961  1 ! SIDE EFFECTS:
;  306     2962  1 !
;  307     2963  1 !       The specified addresses have their values changed.
```

```
;  308     2964  1 !       If a failure in a write occurs, the routine SIGNALs an error.
;  309     2965  1 !
;  310     2966  1 !--
;  311     2967  1
;  312     2968  2 BEGIN
;  313     2969  2
;  314     2970  2 LITERAL
;  315     2971  2         NOP_INSTR       = 1,                            ! Fill char for instructions
;  316     2972  2         ZERO_BYTE       = 0,                            ! Fill char for data
;  317     2973  2         ONE_PAGE        = 1,                            ! Number of pages to expand patch area
;  318     2974  2         MAX_INST_LEN    = 80;                           ! Maximum number of binary bytes in an instr
;  319     2975  2
;  320     2976  2 LOCAL
;  321     2977  2         INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE],
;  322     2978  2         FILL_CHAR: BYTE,                                ! Fill character for delete command
;  323     2979  2         OLD_CONTENTS:   VECTOR[TTY_OUT_WIDTH, BYTE],    ! Buffer to hold old contents of location
;  324     2980  2         MAPPED_LOC,                                     ! Mapped address of deposit destination
;  325     2981  2         UNMAPPED_LOC,                                   ! Unmapped address of deposit destination
;  326     2982  2         ISE_ADDR,                                       ! Image section entry for deposit destinatio
;  327     2983  2         DEP_SRC_ADR,                                    ! Pointer to deposit source
;  328     2984  2         DEP_SIZ,                                        ! Size of deposit to be made
;  329     2985  2         POINTER;
;  330     2986  2
;  331     2987  2 !++
;  332     2988  2 ! Check that all parameters were specified on the command.
;  333     2989  2 !--
;  334     2990  2 POINTER = .PAT$GL_HEAD_LST;
;  335     2991  3 IF (.POINTER EQLA 0) OR (.LIST_ELEM_FLINK(.POINTER) EQLA 0)
;  336     2992  2 THEN
;  337     2993  2         SIGNAL (PAT$_INVCMD);
;  338     2994  2
;  339     2995  2 !++
;  340     2996  2 ! Initialize unmapped address and PAT$GL_CONTEXT.  The context bit causes
;  341     2997  2 ! the routine PAT$OUT_MEM_LOC to display a location instead of evaluating
;  342     2998  2 ! a numeric expression.
;  343     2999  2 !--
;  344     3000  2 UNMAPPED_LOC = .LIST_ELEM_EXP1 (.POINTER);
;  345     3001  2 PAT$GL_CONTEXT[EXAMINE_BIT] = TRUE;
;  346     3002  2
;  347     3003  2 !++
;  348     3004  2 ! For DEPOSIT commands only:
;  349     3005  2 !
;  350     3006  2 ! Check if DEPOSIT qualifier, "/PATCH_AREA", was specified.
;  351     3007  2 ! If so, check that the address specified is identical to the
;  352     3008  2 ! start of the current patch area.  If it is not, report an
;  353     3009  2 ! error and abort the DEPOSIT command.
;  354     3010  2 !--
;  355     3011  2 IF .PAT$GL_CONTEXT [PAT_AREA_BIT]
;  356     3012  2 THEN
;  357     3013  3         BEGIN
;  358     3014  4         IF (.PAT$GL_PATAREA[DSC$W_LENGTH] EQL 0)
;  359     3015  3         THEN
;  360     3016  4                 IF (.PAT$GL_PATAREA [DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
;  361     3017  3                 THEN
;  362     3018  3                         PAT$EXP_AREA(ONE_PAGE);
;  363     3019  4         IF (.PAT$GL_PATAREA [DSC$A_POINTER] NEQA .UNMAPPED_LOC)
;  364     3020  3         THEN
```

```
;  365     3021  3                    SIGNAL(PAT$_NOTPATADR, 2, .PAT$GL_PATAREA[DSC$A_POINTER], .UNMAPPED_LOC);
;  366     3022  2            END;
;  367     3023  2
;  368     3024  2            !++
;  369     3025  2            ! Set the fill character for DELETE commands.
;  370     3026  2            !--
;  371     3027  2            IF (.PAT$GB_MOD_PTR [MODE_INSTRUC])
;  372     3028  2            THEN
;  373     3029  2                    BEGIN
;  374     3030  3                    PAT$GL_SYMTBPTR = .PAT$GL_NEWLABLS;                     ! Use new contents label table
;  375     3031  3                    FILL_CHAR = NOP_INSTR;                                  ! Fill character for instructions
;  376     3032  3                    END
;  377     3033  2            ELSE
;  378     3034  2                    FILL_CHAR = ZERO_BYTE;                                  ! Fill character for data
;  379     3035  2
;  380     3036  2            !++
;  381     3037  2            ! Loop to DEPOSIT (DELETE) all parameters specified in the command.
;  382     3038  2            !--
;  383     3039  2            REPEAT
;  384     3040  3                    BEGIN
;  385     3041  3                    POINTER = .LIST_ELEM_FLINK (.POINTER);
;  386     3042  3
;  387     3043  3                    !++
;  388     3044  3                    ! Now determine the length of the instruction or data
;  389     3045  3                    ! which is to be deposited or deleted.
;  390     3046  3                    !--
;  391     3047  3                    IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
;  392     3048  3                    THEN
;  393     3049  4                            BEGIN
;  394     3050  4                            !++
;  395     3051  4                            ! This is a symbolic instruction to be deposited or deleted.
;  396     3052  4                            ! It is currently in the form of a counted ASCII string that
;  397     3053  4                            ! must be translated into binary form. The call to PAT$INS_ENCODE
;  398     3054  4                            ! needs the address for which the instruction is encoded in
;  399     3055  4                            ! order to resolve branches correctly.
;  400     3056  4                            !--
;  401     3057  4                            IF NOT PAT$INS_ENCODE (.LIST_ELEM_EXP1 (.POINTER),
;  402     3058  4                                    INSTRUC_BUF, .UNMAPPED_LOC,
;  403     3059  5                                    (IF .PAT$GL_CONTEXT[DELETE_BIT]
;  404     3060  5                                            THEN PAT$GL_OLD_ASD
;  405     3061  4                                            ELSE PAT$GL_NEW_ASD),
;  406     3062  4                                    PAT$GL_TEMP_BUF)
;  407     3063  4                            THEN
;  408     3064  4                                    SIGNAL (PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER)); ! This instruction is invalid.
;  409     3065  4                            DEP_SRC_ADR = INSTRUC_BUF [1];
;  410     3066  4                            DEP_SIZ = .INSTRUC_BUF [0];
;  411     3067  4                            END
;  412     3068  3                    ELSE
;  413     3069  4                            BEGIN
;  414     3070  4                            !++
;  415     3071  4                            ! Determine length and address for deposits or deletes which are
;  416     3072  4                            ! not symbolic instructions.  Then check for truncation of new value.
;  417     3073  4                            !--
;  418     3074  4                            DEP_SRC_ADR = LIST_ELEM_EXP1 (.POINTER);
;  419     3075  4                            DEP_SIZ = .PAT$GB_MOD_PTR [MODE_LENGTH];
;  420     3076  5                            IF T.LIST_ELEM_EXP1(.POINTER) LSS 0)
;  421     3077  4                            THEN
```

```
;  422     3078  5                    BEGIN
;  423     3079  5                    IF .(LIST_ELEM_EXP1(.POINTER))<0, .DEP_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
;  424     3080                       THEN
;  425     3081                           SIGNAL(PAT$_NUMTRUNC);
;  426     3082                       END
;  427     3083                ELSE
;  428     3084                       IF .(LIST_ELEM_EXP1(.POINTER))<0, .DEP_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
;  429     3085                       THEN
;  430     3086                           SIGNAL(PAT$_NUMTRUNC);
;  431     3087                ELSE;
;  432     3088
;  433     3089            !++
;  434     3090            ! Now write the new values into a temporary buffer.  They are not
;  435     3091            ! written directly into memory in case part of the command fails.
;  436     3092            !--
;  437     3093            PAT$FILL_BUF (PAT$GL_TEMP_BUF, .DEP_SRC_ADR, .DEP_SIZ);
;  438     3094
;  439     3095            !++
;  440     3096            ! Finished with current value.  Reset last location,
;  441     3097            ! next location, and last value, and exitloop.
;  442     3098            !--
;  443     3099            PAT$GL_LAST_LOC = .UNMAPPED_LOC;
;  444     3100            UNMAPPED_LOC = .UNMAPPED_LOC + .DEP_SIZ;
;  445     3101            IF NOT .PAT$GB_MOD_PTR [MODE_INSTRUC]
;  446     3102            THEN
;  447     3103                PAT$GL_LAST_VAL = .LIST_ELEM_EXP1 (.POINTER);
;  448     3104
;  449     3105            !++
;  450     3106            ! If there are no more values, then exit loop which builds
;  451     3107            ! temporary deposit buffer.
;  452     3108            !--
;  453     3109  4         IF (.LIST_ELEM_FLINK (.POINTER) EQLA 0)
;  454     3110  3         THEN
;  455     3111  3             EXITLOOP;
;  456     3112  2         END;
;  457     3113  2
;  458     3114  2     !++
;  459     3115  2     ! For DEPOSIT command only:
;  460     3116  2     !
;  461     3117  2     ! First check if this is writing into the patch area.  If so, check that there
;  462     3118  2     ! is enough room in the patch area.  If not, then expand the patch area if
;  463     3119  2     ! possible (that is, if the current patch area is the one defined int the image
;  464     3120  2     ! header).  Otherwise, report an error and abort this command.
;  465     3121  2     !--
;  466     3122  2     IF .PAT$GL_CONTEXT[PAT_AREA_BIT]
;  467     3123  2     THEN
;  468     3124  3         BEGIN
;  469     3125  4         IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
;  470     3126  3         THEN
;  471     3127  4             BEGIN
;  472     3128  5             IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
;  473     3129  4             THEN
;  474     3130  5                 PAT$EXP_AREA((.PAT$GL_TEMP_BUF[DSC$W_LENGTH] +
;  475     3131  4                                 A_PAGE - 1)/A_PAGE)
;  476     3132  4             ELSE
;  477     3133  4                 SIGNAL(PAT$_INSUFPAT, 2, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
;  478     3134  4                                 .PAT$GL_PATAREA[DSC$A_POINTER],
```

```
;  479     3135  4                                   .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  480     3136  4                       END;
;  481     3137  3               END;
;  482     3138  3
;  483     3139  3           !++
;  484     3140  3           ! Now resolve any forward references inside the symbolic instruction operands.
;  485     3141  3           !--
;  486     3142  3           PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
;  487     3143  3
;  488     3144  3           !++
;  489     3145  3           ! Output the old values.
;  490     3146  3           !--
;  491     3147  3           PAT$GL_NEXT_LOC = .LIST_ELEM_EXP1(.PAT$GL_HEAD_LST);
;  492     3148  3           WHILE .PAT$GL_NEXT_LOC [SSA .UNMAPPED_LOC
;  493     3149  3           DO
;  494     3150  3                   PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, CASE_TABLE);
;  495     3151  3
;  496     3152  3           !++
;  497     3153  3           ! For DELETE commands only:
;  498     3154  3           !
;  499     3155  3           ! Verify that the old values were actually in memory for DELETE commands.
;  500     3156  3           ! Then fill the temporary buffer with the appropriate fill character.
;  501     3157  3           !--
;  502     3158  3           IF .PAT$GL_CONTEXT[DELETE_BIT]
;  503     3159  3           THEN
;  504     3160  3                   BEGIN
;  505     3161  3                   !++
;  506     3162  3                   ! Now get the actual value in the location and
;  507     3163  3                   ! check that it equals the specified value.
;  508     3164  3                   !--
;  509     3165  3                   LOCAL
;  510     3166  3                           BYTE_COUNT,                             ! Count of bytes verified
;  511     3167  3                           BUF_SIZE;                               ! Size of old contents buffer to get
;  512     3168  3                   BYTE_COUNT = 0;
;  513     3169  4                   WHILE (.BYTE_COUNT LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
;  514     3170  3                   DO
;  515     3171  4                           BEGIN
;  516     3172  5                           IF ((BUF_SIZE = .PAT$GL_TEMP_BUF[DSC$W_LENGTH] - .BYTE_COUNT) GTR TTY_OUT_WIDTH)
;  517     3173  4                           THEN
;  518     3174  4                                   BUF_SIZE = TTY_OUT_WIDTH;       ! Request only as much as buffer can hold
;  519     3175  4                           PAT$GET_VALUE(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST)+.BYTE_COUNT,
;  520     3176  4                                   .BUF_SIZE, OLD_CONTENTS);
;  521     3177  4                           IF CH$NEQ(.BUF_SIZE, .PAT$GL_TEMP_BUF[DSC$A_POINTER]+.BYTE_COUNT,
;  522     3178  4                                   .BUF_SIZE, OLD_CONTENTS)
;  523     3179  4                           THEN
;  524     3180  4                                   SIGNAL(PAT$_DIFVAL+MSG$K_WARN);
;  525     3181  4                           BYTE_COUNT = .BYTE_COUNT + .BUF_SIZE;
;  526     3182  3                           END;
;  527     3183  3                   CH$FILL(.FILL_CHAR, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
;  528     3184  3                           .PAT$GL_TEMP_BUF[DSC$A_POINTER]);
;  529     3185  2                   END;
;  530     3186  2
;  531     3187  2           !++
;  532     3188  2           ! Now write the temporary buffer into memory.
;  533     3189  2           !--
;  534     3190  2           PAT$GL_NEXT_LOC = .LIST_ELEM_EXP1(.PAT$GL_HEAD_LST);
;  535     3191  2           PAT$WRITE_MEM (.PAT$GL_NEXT_LOC, .PAT$GL_TEMP_BUF[DSC$A_POINTER],
```

```
;  536     3192  2                          .PAT$GL_TEMP_BUF[DSC$W_LENGTH]);
;  537     3193  2
;  538     3194  2              !++
;  539     3195  2              ! Output the new values.
;  540     3196  2              !--
;  541     3197  2              WHILE .PAT$GL_NEXT_LOC LSSA .UNMAPPED_LOC
;  542     3198  2              DO
;  543     3199  2                  PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, CASE_TABLE);
;  544     3200  2              !++
;  545     3201  2              ! Now check if the deposit was into the current patch area.  If so,
;  546     3202  2              ! update the patch area descriptor.
;  547     3203  2              !--
;  548     3204  2              IF .PAT$GL_CONTEXT [PAT_AREA_BIT]
;  549     3205  2              THEN
;  550     3206  3                  BEGIN
;  551     3207  3                  PAT$GL_PATAREA[DSC$A_POINTER] = .PAT$GL_PATAREA[DSC$A_POINTER] +
;  552     3208  3                                                  .PAT$GL_TEMP_BUF[DSC$W_LENGTH];
;  553     3209  3                  PAT$GL_PATAREA[DSC$W_LENGTH] = .PAT$GL_PATAREA[DSC$W_LENGTH] -
;  554     3210  3                                                  .PAT$GL_TEMP_BUF[DSC$W_LENGTH];
;  555     3211  2                  END;
;  556     3212  2              !++
;  557     3213  2              ! Now add the new labels to the user-defined symbol table.
;  558     3214  2              !--
;  559     3215  2              PAT$ADD_LABELS(PAT$GL_NEWLABLS);
;  560     3216  2
;  561     3217  2
;  562     3218  2              RETURN;
;  563     3219  1              END;


                                        .TITLE  PATEXA
                                        .IDENT  \V04-000\

                                        .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                    02  00000 P.AAA:   .BYTE   2
                30  52  00001          .ASCII  \R0\                                    ;
                    00  00003          .BYTE   0
                    02  00004          .BYTE   2
                31  52  00005          .ASCII  \R1\                                    ;
                    00  00007          .BYTE   0
                    02  00008          .BYTE   2
                32  52  00009          .ASCII  \R2\                                    ;
                    00  0000B          .BYTE   0
                    02  0000C          .BYTE   2
                33  52  0000D          .ASCII  \R3\                                    ;
                    00  0000F          .BYTE   0
                    02  00010          .BYTE   2
                34  52  00011          .ASCII  \R4\                                    ;
                    00  00013          .BYTE   0
                    02  00014          .BYTE   2
                35  52  00015          .ASCII  \R5\                                    ;
                    00  00017          .BYTE   0
                    02  00018          .BYTE   2
                36  52  00019          .ASCII  \R6\                                    ;
                    00  0001B          .BYTE   0
                    02  0001C          .BYTE   2                                       ;
```

```
                                                ;*****************************************************************************
                                                ;*                                                                           *
                                                ;*  Copyright (c) 1978, 1979, 1980, 1981, 1982, 1983, 1984 by                *
                                                ;*  Digital Equipment Corporation, Maynard, Massachusetts.                   *
                                                ;*                                                                           *
                                                ;*  This software is furnished under a license and may be used and  copied  *
                                                ;*  only  in  accordance  with  the  terms  of  such  license  and with the  *
                                                ;*  inclusion of the above copyright notice.  This software or  any  other  *
                                                ;*  copies  thereof may not be provided or otherwise made available to any   *
                                                ;*  other person.  No title to and ownership of the  software  is  hereby   *
                                                ;*  transferred.                                                             *
                                                ;*                                                                           *
                                                ;*  The information in this software is subject to change  without  notice   *
                                                ;*  and  should  not  be  construed  as  a commitment by Digital Equipment   *
                                                ;*  Corporation.                                                             *
                                                ;*                                                                           *
                                                ;*  Digital assumes no responsibility for the use or  reliability  of  its   *
                                                ;*  software on equipment which is not supplied by Digital.                  *
                                                ;*                                                                           *
                                                ;*****************************************************************************

                                                ;++
                                                ; FACILITY:  PATCH
                                                ;
                                                ; ABSTRACT:
                                                ;
                                                ;       This module contains the data structures for the PATCH utility.
                                                ;
                                                ; ENVIRONMENT:
                                                ;
                                                ;       VAX/VMS operating system
                                                ;
                                                ; AUTHOR:  Tim Halvorsen, 1-Aug-1980
                                                ;
                                                ; MODIFIED BY:
                                                ;
                                                ;       V03-001 TMH0001 Tim Halvorsen   14-Sep-1982
                                                ;               Add support for new PATCH commands.
                                                ;--

                                                ;
                                                ; Define register table
                                                ;

                                                REGISTER_TABLE:
                                                        .ASCII  \R0\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R1\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R2\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R3\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R4\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R5\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R6\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R7\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R8\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \R9\
                                                        .BYTE   0
                                                        .BYTE   3
                                                        .ASCII  \R10\
                                                        .BYTE   3
                                                        .ASCII  \R11\
                                                        .BYTE   2
                                                        .ASCII  \AP\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \FP\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \SP\
                                                        .BYTE   0
                                                        .BYTE   2
                                                        .ASCII  \PC\
                                                        .BYTE   0
                                                        .BYTE   3
                                                        .ASCII  \PSL\
00 00 27 44 41 21 27 05                         P.AAB:  .ASCII  <5>\'!AD'\<0><0>
                20 20 3A 03                     P.AAC:  .ASCII  <3>\:  \
                      00 2F 21 02               P.AAD:  .ASCII  <2>\!/\<0>
00 00 09 3A 64 6C 6F 05                         P.AAE:  .ASCII  <5>\old:\<9><0><0>
00 00 09 3A 77 65 6E 05                         P.AAF:  .ASCII  <5>\new:\<9><0><0>

                                                ;
                                                ; Define size of each table entry
                                                ;

                                                ISE$C_SIZE==            20
                                                TXT$C_SIZE==            4
                                                PAL$C_SIZE==            16
                                                ASD$C_SIZE==            9
                                                FWR$C_SIZE==            24

                                                ;
                                                ; Define pointers to each table
                                                ;

                                                REGISTER_TABLE=         P.AAA
                                                CS_ASCII=               P.AAB
                                                COLON_TAB_STG=          P.AAC
                                                CAR_CTL_STG=            P.AAD
                                                OLD_TAB_STG=            P.AAE
                                                NEW_TAB_STG=            P.AAF

                                                ;
                                                ; Define external symbols
                                                ;

                                                        .EXTRN  PAT$FAO_OUT, PAT$GB_SYMBOLS
                                                        .EXTRN  PAT$GL_OLD_ASD, PAT$GL_NEW_ASD
                                                        .EXTRN  PAT$GL_TEMP_BUF
                                                        .EXTRN  PAT$GL_RLOC_BUF
                                                        .EXTRN  PAT$GB_SUBST_IN
                                                        .EXTRN  PAT$GL_BR_DISPL
                                                        .EXTRN  PAT$GL_PATAREA, PAT$GL_IMGHDR
                                                        .EXTRN  PAT$GB_LOC_TYPE
                                                        .EXTRN  PAT$GB_MOD_PTR, PAT$GL_IHPPTR
                                                        .EXTRN  PAT$CP_OUT_STR, PAT$GL_CONTEXT
                                                        .EXTRN  PAT$GL_BUF_SIZ, PAT$GL_HEAD_LST
                                                        .EXTRN  PAT$GL_LAST_LOC
                                                        .EXTRN  PAT$GL_LAST_VAL
                                                        .EXTRN  PAT$GL_NEXT_LOC
```

```
                                        .EXTRN  PAT$GL_SYMTBPTR
                                        .EXTRN  PAT$GL_OLDLABLS
                                        .EXTRN  PAT$GL_NEWLABLS
                                        .EXTRN  PAT$GL_RLCLABLS
                                        .EXTRN  PAT$ADD_LABELS, PAT$ADD_NT_T_PV
                                        .EXTRN  PAT$EXP_AREA, PAT$FAO_PUT
                                        .EXTRN  PAT$FREERELEASE
                                        .EXTRN  PAT$FREEZ, PAT$GET_NXT_LVT
                                        .EXTRN  PAT$GET_VALUE, PAT$INS_DECODE
                                        .EXTRN  PAT$INS_ENCODE, PAT$MAP_ADDR
                                        .EXTRN  PAT$OUT_NUM_VAL
                                        .EXTRN  PAT$OUT_PUT, PAT$OUT_SYM_VAL
                                        .EXTRN  PAT$PRINT_PATH, PAT$RESOLVE_INS
                                        .EXTRN  PAT$SYMBOL_VALU
                                        .EXTRN  PAT$UNMAP_ADDR, PAT$VAL_TO_SYM
                                        .EXTRN  PAT$WRITE_MEM
                                        .WEAK   ACCESS_CHECK

                                        .PSECT  _PAT$CODE,NOWRT,2

                        0FFC 00000      .ENTRY  PAT$DEPOSIT_CMD, Save R2,R3,R4,R5,R6,R7,R8,-; 2908
                                                R9,R10,R11
5B 00000000G   00   9E 00002            MOVAB   LIB$SIGNAL, R11                             ;
5A 00000000G   EF   9E 00009            MOVAB   PAT$GL_NEXT_LOC, R10                        ;
59 00000000G   EF   9E 00010            MOVAB   PAT$GL_PATAREA, R9                          ;
58 00000000G   EF   9E 00017            MOVAB   PAT$GL_TEMP_BUF, R8                         ;
5E      FF2C   CE   9E 0001E            MOVAB   -212(SP), SP                                ;
52 00000000G   EF   D0 00023            MOVL    PAT$GL_HEAD_LST, POINTER                    ; 2990
                   04   13 0002A        BEQL    1$                                          ; 2991
                   62   D5 0002C        TSTL    (POINTER)                                   ;
                   09   12 0002E        BNEQ    2$                                          ;
                006D80DA   8F   DD 00030 1$:     PUSHL   #7176410                            ; 2993
                      6B   01   FB 00036        CALLS   #1, LIB$SIGNAL                      ;
                      56   04   A2   D0 00039 2$:     MOVL    4(POINTER), UNMAPPED_LOC    ; 3000
          00000000G   EF        01   88 0003D        BISB2   #1, PAT$GL_CONTEXT+1            ; 3001
    3B 00000000G   EF        03   E1 00044        BBC     #3, PAT$GL_CONTEXT+2, 4$        ; 3011
                   00   B9   B5 0004C            TSTW    @PAT$GL_PATAREA                 ; 3014
                        1A   12 0004F            BNEQ    3$                              ;
                      51   69   D0 00051        MOVL    PAT$GL_PATAREA, R1                  ; 3016
                      50 00000000G   EF   D0 00054        MOVL    PAT$GL_IHPPTR, R0       ;
                14    A0   04   A1   D1 0005B        CMPL    4(R1), 20(R0)               ;
                        09   12 00060            BNEQ    3$                              ;
                        01   DD 00062            PUSHL   #1                              ; 3018
          00000000G   EF        01   FB 00064        CALLS   #1, PAT$EXP_AREA            ;
                      50   69   D0 0006B 3$:     MOVL    PAT$GL_PATAREA, R0              ; 3019
                      56   04   A0   D1 0006E        CMPL    4(R0), UNMAPPED_LOC         ;
                        13   13 00072            BEQL    4$                              ;
                        56   DD 00074            PUSHL   UNMAPPED_LOC                    ; 3021
                      50   69   D0 00076        MOVL    PAT$GL_PATAREA, R0              ;
                   04   A0   DD 00079            PUSHL   4(R0)                           ;
                        02   DD 0007C            PUSHL   #2                              ;
                006D8122   8F   DD 0007E        PUSHL   #7176482                        ;
                      6B   04   FB 00084        CALLS   #4, LIB$SIGNAL                  ;
                      50 00000000G   EF   D0 00087 4$:     MOVL    PAT$GB_MOD_PTR, R0      ; 3027
                      10   03   A0   E9 0008E        BLBC    3(R0), 5$                   ;
          00000000G   EF 00000000G   EF   D0 00092        MOVL    PAT$GL_NEWLABLS, PAT$GL_SYMTBPTR ; 3030
                      57        01   90 0009D        MOVB    #1, FICL_CHAR               ; 3031
```

```
                                                02 11 000A0          BRB     6$                                      ; 3027
                                                57 94 000A2 5$:      CLRB    FILL_CHAR                               ; 3034
                                52              62 D0 000A4 6$:      MOVL    (POINTER), POINTER                      ; 3041
                                50 00000000G    EF D0 000A7          MOVL    PAT$GB_MOD_PTR, R0                      ; 3047
                                46        03    A0 E9 000AE          BLBC    3(R0), 10$
                                                58 DD 000B2          PUSHL   R8                                      ; 3057
                     09 00000000G   EF          06 E1 000B4          BBC     #6, PAT$GL_CONTEXT+2, 7$                ; 3059
                                50 00000000G    EF 9E 000BC          MOVAB   PAT$GL_OLD_ASD, R0
                                                07 11 000C3          BRB     8$
                                50 00000000G    EF 9E 000C5 7$:      MOVAB   PAT$GL_NEW_ASD, R0
                                                50 DD 000CC 8$:      PUSHL   R0
                                                56 DD 000CE          PUSHL   UNMAPPED_LOC                            ; 3058
                                          B0    AD 9F 000D0          PUSHAB  INSTRUC_BUF                             ; 3057
                                          04    A2 DD 000D3          PUSHL   4(POINTER)
                          00000000G   EF        05 FB 000D6          CALLS   #5, PAT$INS_ENCODE
                                      0E        50 E8 000DD          BLBS    R0, 9$
                                          04    A2 DD 000E0          PUSHL   4(POINTER)                              ; 3064
                                                01 DD 000E3          PUSHL   #1
                                    006D810A    8F DD 000E5          PUSHL   #7176458
                                52              03 FB 000EB          CALLS   #3, LIB$SIGNAL
                                54        B1    AD 9E 000EE 9$:      MOVAB   INSTRUC_BUF+1, DEP_SRC_ADR              ; 3065
                                53        B0    AD 9A 000F2          MOVZBL  INSTRUC_BUF, DEP_SIZ                    ; 3066
                                                39 11 000F6          BRB     13$                                     ; 3047
                                54        04    A2 9E 000F8 10$:     MOVAB   4(R2), DEP_SRC_ADR                      ; 3074
                                50 00000000G    EF D0 000FC          MOVL    PAT$GB_MOD_PTR, R0                      ; 3075
                                53        01    A0 9A 00103          MOVZBL  1(R0), DEP_SIZ
                                          04    A2 D5 00107          TSTL    4(POINTER)                              ; 3076
                                                0C 18 0010A          BGEQ    11$
                          50                    53 03 78 0010C       ASHL    #3, DEP_SIZ, R0                         ; 3079
          51        04    A2                    50 00 EE 00110       EXTV    #0, R0, 4(POINTER), R1
                                                0A 11 00116          BRB     12$
                          50                    53 03 78 00118 11$:  ASHL    #3, DEP_SIZ, R0                         ; 3084
          51        04    A2                    50 00 EF 0011C       EXTZV   #0, R0, 4(POINTER), R1
                                    04    A2    51 D1 00122 12$:     CMPL    R1, 4(POINTER)
                                                09 13 00126          BEQL    13$
                                    006D8023    8F DD 00128          PUSHL   #7176227                                ; 3086
                                52              01 FB 0012E          CALLS   #1, LIB$SIGNAL
                                                53 DD 00131 13$:     PUSHL   DEP_SIZ                                 ; 3093
                                                54 DD 00133          PUSHL   DEP_SRC_ADR
                                                58 DD 00135          PUSHL   R8
                     00000000V   EF             03 FB 00137          CALLS   #3, PAT$FILL_BUF
                     00000000G   EF             56 D0 0013E          MOVL    UNMAPPED_LOC, PAT$GL_LAST_LOC           ; 3099
                                56              53 C0 00145          ADDL2   DEP_SIZ, UNMAPPED_LOC                   ; 3100
                                50 00000000G    EF D0 00148          MOVL    PAT$GB_MOD_PTR, R0                      ; 3101
                                08        03    A0 E8 0014F          BLBS    3(R0), 14$
                     00000000G   EF        04    A2 D0 00153          MOVL    4(POINTER), PAT$GL_LAST_VAL             ; 3103
                                                62 D5 0015B 14$:     TSTL    (POINTER)                               ; 3109
                                                03 13 0015D          BEQL    15$
                                              FF42 31 0015F          BRW     6$
                     48 00000000G   EF             03 E1 00162 15$:     BBC     #3, PAT$GL_CONTEXT+2, 17$           ; 3122
                                68        00    B9 B1 0016A          CMPW    @PAT$GL_PATAREA, PAT$GL_TEMP_BUF        ; 3125
                                                42 1E 0016E          BGEQU   17$
                                51              69 D0 00170          MOVL    PAT$GL_PATAREA, R1                      ; 3128
                                50 00000000G    EF D0 00173          MOVL    PAT$GL_IHPPTR, R0
                          14    A0        04    A1 D1 0017A          CMPL    4(R1), 20(R0)
                                                19 12 0017F          BNEQ    16$
                                50              68 3C 00181          MOVZWL  PAT$GL_TEMP_BUF, R0                     ; 3130
```

```
                        50     01FF   C0  9E 00184          MOVAB   511(R0), R0
                    7E  50 00000200   8F  C7 00189          DIVL3   #512, R0, -(SP)                                 ;3131
          00000000G EF               01  FB 00191          CALLS   #1, PAT$EXP_AREA
                                     18  11 00198          BRB     17$                                             ;3130
                        7E       00   B9  3C 0019A 16$:     MOVZWL  @PAT$GL_PATAREA, -(SP)                          ;3135
                        50             69  D0 0019E          MOVL    PAT$GL_PATAREA, R0                              ;3134
                                 04    A0  DD 001A1          PUSHL   4(R0)
                        7E             68  3C 001A4          MOVZWL  PAT$GL_TEMP_BUF, -(SP)                          ;3133
                                       02  DD 001A7          PUSHL   #2
                           006D80C2    8F  DD 001A9          PUSHL   #7176386
                        6B             05  FB 001AF          CALLS   #5, LIB$SIGNAL
                                       58  DD 001B2 17$:     PUSHL   R8                                              ;3142
          00000000G     EF             01  FB 001B4          CALLS   #1, PAT$RESOLVE_INS
                        50 00000000G   EF  D0 001BB          MOVL    PAT$GL_HEAD_LST, R0                             ;3147
                        6A       04    A0  D0 001C2          MOVL    4(R0), PAT$GL_NEXT_LOC
                        56             6A  D1 001C6 18$:     CMPL    PAT$GL_NEXT_LOC, UNMAPPED_LOC                   ;3148
                                       19  1E 001C9          BGEQU   19$                                             ;3150
                                       01  DD 001CB          PUSHL   #1
                           00000000G   EF  9F 001CD          PUSHAB  PAT$GL_OLD_ASD
                           00000000'   EF  9F 001D3          PUSHAB  OLD_TAB_STG
                                       6A  DD 001D9          PUSHL   PAT$GL_NEXT_LOC
          00000000V     EF             04  FB 001DB          CALLS   #4, PAT$OUT_MEM_LOC
                                       E2  11 001E2          BRB     18$
                    4F 00000000G EF    06  E1 001E4 19$:     BBC     #6, PAT$GL_CONTEXT+2, 24$                       ;3158
                                       54  D4 001EC          CLRL    BYTE_COUNT                                      ;3168
                54  68             10    00  ED 001EE 20$:     CMPZV   #0, #16, PAT$GL_TEMP_BUF, BYTE_COUNT        ;3169
                                       3F  15 001F3          BLEQ    23$
                        55             68  3C 001F5          MOVZWL  PAT$GL_TEMP_BUF, BUF_SIZE                       ;3172
                        55             54  C2 001F8          SUBL2   BYTE_COUNT, BUF_SIZE
          00000084      8F             55  D1 001FB          CMPL    BUF_SIZE, #132
                                       04  15 00202          BLEQ    21$
                        55       84    8F  9A 00204          MOVZBL  #132, BUF_SIZE                                  ;3174
                              4020     8F  BB 00208 21$:     PUSHR   #^M<R5,SP>                                      ;3176
                        50 00000000G   EF  D0 0020C          MOVL    PAT$GL_HEAD_LST, R0                             ;3175
                                 04 B044   9F  00213          PUSHAB  @4(R0)[BYTE_COUNT]
          00000000G     EF             03  FB 00217          CALLS   #3, PAT$GET_VALUE
                    6E        04 B844    55  29 0021E          CMPC3   BUF_SIZE, @PAT$GL_TEMP_BUF+4[BYTE_COUNT], -    ;3177
                                                                OLD_CONTENTS
                                       09  13 00224          BEQL    22$                                             ;3180
                           006D8290    8F  DD 00226          PUSHL   #7176848
                        6B             01  FB 0022C          CALLS   #1, LIB$SIGNAL
                        54             55  C0 0022F 22$:     ADDL2   BUF_SIZE, BYTE_COUNT                            ;3181
                                       BA  11 00232          BRB     20$                                             ;3169
                57      6E             00  2C 00234 23$:     MOVC5   #0, (SP), FILL_CHAR, PAT$GL_TEMP_BUF, -         ;3184
                              04       B8    00239                  @PAT$GL_TEMP_BUF+4
                        50 00000000G   EF  D0 0023B 24$:     MOVL    PAT$GL_READ_LST, R0                             ;3190
                        6A       04    A0  D0 00242          MOVL    4(R0), PAT$GL_NEXT_LOC
                        7E             68  3C 00246          MOVZWL  PAT$GL_TEMP_BUF, -(SP)                          ;3192
                                 04    A8  DD 00249          PUSHL   PAT$GL_TEMP_BUF+4                               ;3191
                                       6A  DD 0024C          PUSHL   PAT$GL_NEXT_LOC
          00000000G     EF             03  FB 0024E          CALLS   #3, PAT$WRITE_MEM
                        56             6A  D1 00255 25$:     CMPL    PAT$GL_NEXT_LOC, UNMAPPED_LOC                   ;3197
                                       19  1E 00258          BGEQU   26$
                                       01  DD 0025A          PUSHL   #1                                              ;3199
                           00000000G   EF  9F 0025C          PUSHAB  PAT$GL_NEW_ASD
                           00000000'   EF  9F 00262          PUSHAB  NEW_TAB_STG
                                       6A  DD 00268          PUSHL   PAT$GL_NEXT_LOC
```

```
                    00000000V  EF              04  FB 0026A             CALLS   #4, PAT$OUT_MEM_LOC                                  ;
                               E2              11  00271                BRB     25$                                                 ;
                 13 00000000G  EF              03  E1 00273 26$:        BBC     #3, PAT$GL_CONTEXT+2, 27$                           ; 3204
                               50              69  D0 0027B             MOVL    PAT$GL_PATAREA, R0                                  ; 3207
                               51              69  D0 0027E             MOVL    PAT$GL_PATAREA, R1                                  ;
                               52              68  3C 00281             MOVZWL  PAT$GL_TEMP_BUF, R2                                 ; 3208
                           04  A0          04 B142  9E 00284             MOVAB   @4(R1)[R2], 4(R0)                                   ;
                           00  B9              68  A2 0028A             SUBW2   PAT$GL_TEMP_BUF, @PAT$GL_PATAREA                    ; 3210
                                    00000000G  EF  9F 0028E 27$:        PUSHAB  PAT$GL_NEWLABLS                                     ; 3215
                    00000000G  EF              01  FB 00294             CALLS   #1, PAT$ADD_LABELS                                  ;
                                               04  0029B                 RET                                                         ; 3219

; Routine Size:  668 bytes,    Routine Base:  _PAT$CODE + 0000

;  564          3220  1 SWITCHES OPTIMIZE;
```

```
;   566          3221  1 GLOBAL ROUTINE PAT$EXAMINE_CMD : NOVALUE =
;   567          3222  1
;   568          3223  1 !++
;   569          3224  1 ! FUNCTIONAL DESCRIPTION:
;   570          3225  1 !
;   571          3226  1 !       Examines a list of addresses.
;   572          3227  1 !
;   573          3228  1 ! CALLING SEQUENCE:
;   574          3229  1 !
;   575          3230  1 !       PAT$EXAMINE_CMD ()
;   576          3231  1 !
;   577          3232  1 ! INPUTS:
;   578          3233  1 !
;   579          3234  1 !       none
;   580          3235  1 !
;   581          3236  1 ! IMPLICIT INPUTS:
;   582          3237  1 !
;   583          3238  1 !       The address of the first element of a list of addresses.
;   584          3239  1 !       The last address examined, and the next logical address to examine.
;   585          3240  1 !
;   586          3241  1 ! OUTPUTS:
;   587          3242  1 !
;   588          3243  1 !       none
;   589          3244  1 !
;   590          3245  1 ! IMPLICIT OUTPUTS:
;   591          3246  1 !
;   592          3247  1 !       New values for last and next location, and last value
;   593          3248  1 !
;   594          3249  1 ! ROUTINE VALUE:
;   595          3250  1 !
;   596          3251  1 !       novalue
;   597          3252  1 !
;   598          3253  1 ! SIDE EFFECTS:
;   599          3254  1 !
;   600          3255  1 !       The values of various addresses are output.
;   601          3256  1 !       If an error occurs, the routine returns without further
;   602          3257  1 !       processing except to output an error message to the output
;   603          3258  1 !       device.
;   604          3259  1 !
;   605          3260  1 !--
;   606          3261  1
;   607          3262  2 BEGIN
;   608          3263  2
;   609          3264  2 LOCAL
;   610          3265  2         MAPPED_NEXT_LOC,                              ! Mapped address of next location
;   611          3266  2         ISE_ADDR,                                     ! ISE address for mapped address
;   612          3267  2         POINTER;
;   613          3268  2
;   614          3269  2 POINTER = .PAT$GL_HEAD_LST;
;   615          3270  3 IF (.POINTER EQL 0)
;   616          3271  2 THEN
;   617          3272  2
;   618          3273  2         !++
;   619          3274  2         ! No location was specified. Examine the next location in sequence.
;   620          3275  2         !--
;   621          3276  2         PAT$OUT_MEM_LOC (.PAT$GL_NEXT_LOC, 0, PAT$GL_OLD_ASD, CASE_TABLE)
;   622          3277  2 ELSE DO
```

```
;  623     3278  3          BEGIN
;  624     3279
;  625     3280  3          LOCAL
;  626     3281  3                  LAST_LOC;
;  627     3282
;  628     3283  3          !++
;  629     3284  3          ! Pick up the next value which we will try to
;  630     3285  3          ! display and copy it into LAST_LOC.
;  631     3286  3          !--
;  632     3287  3          LAST_LOC = .LIST_ELEM_EXP1 (.POINTER);
;  633     3288  3
;  634     3289  3          !++
;  635     3290  3          ! If the end range argument is null, then make it the same as the start
;  636     3291  3          ! range argument so that only one location will be displayed.
;  637     3292  3          !--
;  638     3293  3          IF .LIST_ELEM_EXP2 (.POINTER) EQL 0
;  639     3294  3          THEN LIST_ELEM_EXP2 (.POINTER) = .LIST_ELEM_EXP1 (.POINTER);
;  640     3295  3
;  641     3296  3          !++
;  642     3297  3          ! Check for range reversal.
;  643     3298  3          !--
;  644     3299  4          IF( .LIST_ELEM_EXP2(.POINTER) LSSA .LIST_ELEM_EXP1(.POINTER) )
;  645     3300  3          THEN
;  646     3301  4                  BEGIN
;  647     3302  4                  SIGNAL (PAT$_EXARANGE);
;  648     3303  4                  RETURN;
;  649     3304  4                  END;
;  650     3305  3          WHILE (.LAST_LOC LEQ .LIST_ELEM_EXP2 (.POINTER)) DO
;  651     3306  4                  BEGIN
;  652     3307  4                  IF NOT PAT$OUT_MEM_LOC (.LAST_LOC, 0, PAT$GL_OLD_ASD, CASE_TABLE)
;  653     3308  4                  THEN RETURN;
;  654     3309  4                  LAST_LOC = .PAT$GL_NEXT_LOC;
;  655     3310  3                  END;
;  656     3311  3              END
;  657     3312  2      UNTIL (POINTER = .LIST_ELEM_FLINK (.POINTER)) EQL 0;
;  658     3313  2
;  659     3314  1      END;


                                    007C 00000          .ENTRY  PAT$EXAMINE_CMD, Save R2,R3,R4,R5,R6          ; 3221
                56 00000000V    EF  9E 00002            MOVAB   PAT$OUT_MEM_LOC, R6                           ;
                55 00000000G    EF  9E 00009            MOVAB   PAT$GL_NEXT_LOC, R5                           ;
                54 00000000G    EF  9E 00010            MOVAB   PAT$GL_OLD_ASD, R4                            ;
                52 00000000G    EF  D0 00017            MOVL    PAT$GL_HEAD_LST, POINTER                      ; 3269
                                0C  12 0001E            BNEQ    1$                                            ; 3270
                                01  DD 00020            PUSHL   #1                                            ; 3276
                                54  DD 00022            PUSHL   R4                                            ;
                                7E  D4 00024            CLRL    -(SP)                                         ;
                                65  DD 00026            PUSHL   PAT$GL_NEXT_LOC                               ;
                66              04  FB 00028            CALLS   #4, PAT$OUT_MEM_LOC                           ;
                                    04 0002B            RET                                                   ;
                53      04      A2  D0 0002C    1$:     MOVL    4(POINTER), LAST_LOC                          ; 3287
                        08      A2  D5 00030            TSTL    8(POINTER)                                    ; 3293
                                05  12 00033            BNEQ    2$                                            ;
```

```
                08   A2        04   A2  D0 00035          MOVL    4(POINTER), 8(POINTER)                  ; 3294
                04   A2        08   A2  D1 0003A 2$:      CMPL    8(POINTER), 4(POINTER)                  ; 3299
                                     0E  1E 0003F          BGEQU   3$                                      ;
                          006D80AA   8F  DD 00041          PUSHL   #7176362                                ; 3302
          00000000G   00             01  FB 00047          CALLS   #1, LIB$SIGNAL                          ;
                                     04  0004E          RET                                             ; 3301
                08   A2             53  D1 0004F 3$:      CMPL    LAST_LOC, 8(POINTER)                    ; 3305
                                     13  14 00053          BGTR    4$                                      ;
                                     01  DD 00055          PUSHL   #1                                      ; 3307
                                     54  DD 00057          PUSHL   R4                                      ;
                                     7E  D4 00059          CLRL    -(SP)                                   ;
                                     53  DD 0005B          PUSHL   LAST_LOC                                ;
                              66     04  FB 0005D          CALLS   #4, PAT$OUT_MEM_LOC                     ;
                              0A     50  E9 00060          BLBC    R0, 5$                                  ;
                              53     65  D0 00063          MOVL    PAT$GL_NEXT_LOC, LAST_LOC               ; 3309
                                     E7  11 00066          BRB     3$                                      ; 3305
                              52     62  D0 00068 4$:      MOVL    (POINTER), POINTER                      ; 3312
                                     BF  12 0006B          BNEQ    1$                                      ;
                                         04 0006D 5$:      RET                                             ; 3314

; Routine Size:  110 bytes,    Routine Base:  _PAT$CODE + 029C
```

```
;  661     3315  1 GLOBAL ROUTINE PAT$REPLACE_CMD : NOVALUE =
;  662     3316  1
;  663     3317  1 !++
;  664     3318  1 ! FUNCTIONAL DESCRIPTION:
;  665     3319  1 !
;  666     3320  1 !       This routine handles all REPLACE, INSERT and VERIFY commands, those for
;  667     3321  1 !       instructions and those for values.  The instruction/value specified in
;  668     3322  1 !       the command must be identical to those actually contained in the
;  669     3323  1 !       location, otherwise an error message is produced and the command
;  670     3324  1 !       ends prematurely.
;  671     3325  1 !
;  672     3326  1 !       The command argument list is made up of entries that are
;  673     3327  1 !       each three longwords long. The first is a forward link to the
;  674     3328  1 !       next entry. The second longword in the first entry in the list
;  675     3329  1 !       is the address into which some value(s) is (are) to be replaced.
;  676     3330  1 !       The third longword is unused. The second longword in successive
;  677     3331  1 !       arguments is the old values/instructions and thier replacemeents.
;  678     3332  1 !       The first value should be the contents of the specified location;
;  679     3333  1 !       the second, the contents of that location plus the current mode_length,
;  680     3334  1 !       etc.  The last old value has an EXIT_TOKEN in the third longword
;  681     3335  1 !       (all others have zero).
;  682     3336  1 !
;  683     3337  1 !       For instruction replacement, the second longword contains
;  684     3338  1 !       the address of a counted ascii stream that is to be translated
;  685     3339  1 !       into a binary instruction.
;  686     3340  1 !
;  687     3341  1 !       The VERIFY command is identical to the replace command save
;  688     3342  1 !       that it has no replacement values.  The INSERT command has only one
;  689     3343  1 !       old instruction and causes it to be moved to the patch area instead
;  690     3344  1 !       of replaced.
;  691     3345  1 !
;  692     3346  1 !       There are three label tables used to differentiate between old labels,
;  693     3347  1 !       new un-relocated labels, and new relocated labels.  The table(s) used
;  694     3348  1 !       to resolve symbols inside symbolic instructions depends updon which
;  695     3349  1 !       instruction is being encoded (old or new) and whether or not they are
;  696     3350  1 !       being relocated to patch area.  Three tables are necessary to handle
;  697     3351  1 !       relocation correctly, i.e., old labels can be used for all instructions
;  698     3352  1 !       but new labels cannot be used for relocated instructions.  No labels
;  699     3353  1 !       can be added to the user-defined symbol table until the PATCH command
;  700     3354  1 !       is successfully executed.
;  701     3355  1 !
;  702     3356  1 ! CALLING SEQUENCE:
;  703     3357  1 !
;  704     3358  1 !       PAT$REPLACE_CMD ()
;  705     3359  1 !
;  706     3360  1 ! INPUTS:
;  707     3361  1 !
;  708     3362  1 !       none
;  709     3363  1 !
;  710     3364  1 ! IMPLICIT INPUTS:
;  711     3365  1 !
;  712     3366  1 !       PAT$GL_HEAD_LST, the head of the linked PATCH command argument list.
;  713     3367  1 !       the current mode, and the current patch area descriptor.
;  714     3368  1 !
;  715     3369  1 ! OUTPUTS:
;  716     3370  1 !
;  717     3371  1 !       none
```

```
;  718     3372  1 !
;  719     3373  1 ! IMPLICIT OUTPUTS:
;  720     3374  1 !
;  721     3375  1 !       none
;  722     3376  1 !
;  723     3377  1 ! ROUTINE VALUE:
;  724     3378  1 !
;  725     3379  1 !       novalue
;  726     3380  1 !
;  727     3381  1 ! SIDE EFFECTS:
;  728     3382  1 !
;  729     3383  1 !       If a failure in a write or contents verification occurs,
;  730     3384  1 !       the routine returns immediately.  If the command is executed
;  731     3385  1 !       successfully, then the specified addresses have new values and any
;  732     3386  1 !       labels in the command are added to the user-defined symbol table.
;  733     3387  1 !
;  734     3388  1 !--
;  735     3389  1
;  736     3390  2 BEGIN
;  737     3391  2
;  738     3392  2 LITERAL
;  739     3393  2         NOP_INSTR = 1,                          ! Fill character for instructions
;  740     3394  2         ZERO_BYTE = 0,                          ! Fill characters for data
;  741     3395  2         MAX_INST_LEN    = 80;                   ! Maximum number of binary bytes in an instr
;  742     3396  2
;  743     3397  2 LOCAL
;  744     3398  2         BYTE_COUNT,                             ! Count of bytesverified
;  745     3399  2         BUF_SIZE,                               ! Size of OLD_CONTENTS to compare
;  746     3400  2         INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE],
;  747     3401  2         OLD_CONTENTS : VECTOR[TTY_OUT_WIDTH,BYTE],  ! Buffer to hold actual contents of location
;  748     3402  2         UNMAPPED_LOC,                           ! Unmapped address of DEPOSIT destination
;  749     3403  2         OLD_VALUE_PTR : REF VECTOR[,BYTE],      ! Pointer of actual contents of location
;  750     3404  2         VAL_SIZ,                                ! Size of current value/instruction
;  751     3405  2         HOLE_SIZ,                               ! Cumulative size to replace
;  752     3406  2         NEXT_LOC,                               ! Pointer to next consecutive location
;  753     3407  2         FILL_CHAR : BYTE,                       ! Fill character for data/instructions
;  754     3408  2         POINTER,
;  755     3409  2         OLD_INS_SIZ,                            ! Number of bytes of old instruction moved f
;  756     3410  2         NEW_INS_PTR;                            ! Pointer to first new instruction argument
;  757     3411  2
;  758     3412  2 !++
;  759     3413  2 ! Check for required parameter.
;  760     3414  2 !--
;  761     3415  2 POINTER = .PAT$GL_HEAD_LST;
;  762     3416  3 IF   (.POINTER EQLA 0) OR (.LIST_ELEM_FLINK (.POINTER) EQLA 0)
;  763     3417  2 THEN
;  764     3418  2         SIGNAL(PAT$_INVCMD);
;  765     3419  2
;  766     3420  2 !++
;  767     3421  2 ! Set the examine bit for PAT$OUT_MEM_LOC.
;  768     3422  2 !
;  769     3423  2 PAT$GL_CONTEXT [EXAMINE_BIT] = TRUE;
;  770     3424  2 UNMAPPED_LOC = .LIST_ELEM_EXP1 (.POINTER);
;  771     3425  2 NEXT_LOC = .LIST_ELEM_EXPT (.POINTER);
;  772     3426  2 HOLE_SIZ = 0;
;  773     3427  2 PAT$GL_SYMTBPTR = .PAT$GL_OLDLABLS;           ! Use old contents label list
;  774     3428  2
```

```
;  775          3429  2 !++
;  776          3430  2 ! Now loop, validating the old contents are the expected values.
;  777          3431  2 ! The last old value argument contains an EXIT_TOKEN in position LIST_ELEM_EXP2.
;  778          3432  2 !--
;  779          3433  2 REPEAT
;  780          3434  3         BEGIN
;  781          3435  3         POINTER = .LIST_ELEM_FLINK (.POINTER);
;  782          3436  3
;  783          3437  3         !++
;  784          3438  3         ! Now compute the binary instruction stream that should be in
;  785          3439  3         ! the location.
;  786          3440  3         !--
;  787          3441  3         IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
;  788          3442  3         THEN
;  789          3443  4                 BEGIN
;  790          3444  4                 !++
;  791          3445  4                 ! This is a symbolic instruction.  It is currently
;  792          3446  4                 ! in the form of a counted ASCII string that must be translated
;  793          3447  4                 ! into binary form. The call to PAT$INS_ENCODE needs the address
;  794          3448  4                 ! into which the instruction is being deposited in order to
;  795          3449  4                 ! resolve branches correctly.
;  796          3450  4                 !--
;  797          3451  4                 IF NOT PAT$INS_ENCODE (.LIST_ELEM_EXP1 (.POINTER),
;  798          3452  4                         INSTRUC_BUF, .NEXT_LOC, PAT$GL_OLD_ASD, PAT$GL_TEMP_BUF)
;  799          3453  4                 THEN
;  800          3454  4                         SIGNAL (PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER)); ! This instruction is not vali
;  801          3455  4                 OLD_VALUE_PTR = INSTRUC_BUF [1];
;  802          3456  4                 VAL_SIZ = .INSTRUC_BUF [0];
;  803          3457  4                 END
;  804          3458  3         ELSE
;  805          3459  4                 BEGIN
;  806          3460  4                 !++
;  807          3461  4                 ! Value is not an instruction.  Therefore it is on the parse
;  808          3462  4                 ! stack.  The current mode for length tells the number of bytes
;  809          3463  4                 ! of the value.  Set pointer to data and size indicator.  Then
;  810          3464  4                 ! check for a truncation error.
;  811          3465  4                 !--
;  812          3466  4                 OLD_VALUE_PTR = LIST_ELEM_EXP1 (.POINTER);
;  813          3467  4                 VAL_SIZ = .PAT$GB_MOD_PTR [MODE_LENGTH];
;  814          3468  4                 IF .LIST_ELEM_EXPT(.POINTER) LSS 0
;  815          3469  4                 THEN
;  816          3470  5                         BEGIN
;  817          3471  5                         IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
;  818          3472  5                         THEN
;  819          3473  5                                 SIGNAL(PAT$_NUMTRUNC);
;  820          3474  5                         END
;  821          3475  4                 ELSE
;  822          3476  4                         IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
;  823          3477  4                         THEN
;  824          3478  4                                 SIGNAL(PAT$_NUMTRUNC);
;  825          3479  3                 END;
;  826          3480  3
;  827          3481  3         PAT$FILL_BUF(PAT$GL_TEMP_BUF, .OLD_VALUE_PTR, .VAL_SIZ);
;  828          3482  3         HOLE_SIZ = .HOLE_SIZ + .VAL_SIZ;                        ! Add current size to cumulative
;  829          3483  3                                                                 ! Output old contents
;  830          3484  3         PAT$OUT_MEM_LOC(.NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, NO_CASE_TABLE);
;  831          3485  3         NEXT_LOC = .PAT$GL_NEXT_LOC;                            ! Point to next location
```

```
;  832          3486  3                !++
;  833          3487  3                ! Check if this is the last old argument.
;  834          3488  3                !--
;  835          3489  3                IF .LIST_ELEM_EXP2(.POINTER) EQL EXIT_TOKEN
;  836          3490  3                THEN
;  837          3491  3                        EXITLOOP;
;  838          3492  3                END;
;  839          3493  3
;  840          3494  3
;  841          3495  2        !++
;  842          3496  2        ! Resolve any forward references in symbolic instruction operands.
;  843          3497  2        !--
;  844          3498  2        PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
;  845          3499  2
;  846          3500  2        !++
;  847          3501  2        ! Now get the actual values in the locations and
;  848          3502  2        ! check that they equal the specified values.
;  849          3503  2        !--
;  850          3504  2        BYTE_COUNT = 0;
;  851          3505  2        WHILE (.BYTE_COUNT LSS .HOLE_SIZ)
;  852          3506  2        DO
;  853          3507  3                BEGIN
;  854          3508  3                IF ((BUF_SIZE = .HOLE_SIZ - .BYTE_COUNT) GTR TTY_OUT_WIDTH)
;  855          3509  3                THEN
;  856          3510  3                        BUF_SIZE = TTY_OUT_WIDTH;
;  857          3511  3                PAT$GET_VALUE(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST)+.BYTE_COUNT,
;  858          3512  3                                .BUF_SIZE, OLD_CONTENTS);
;  859          3513  3                IF CH$NEQ(.BUF_SIZE, .PAT$GL_TEMP_BUF[DSC$A_POINTER]+.BYTE_COUNT,
;  860          3514  3                                .BUF_SIZE, OLD_CONTENTS)
;  861          3515  3                THEN
;  862          3516  3                        SIGNAL(PAT$_DIFVAL+MSG$K_WARN);
;  863          3517  3                BYTE_COUNT = .BYTE_COUNT + .BUF_SIZE;
;  864          3518  2                END;
;  865          3519  2
;  866          3520  2        !++
;  867          3521  2        ! Release the storage holding the old instructions.
;  868          3522  2        !--
;  869          3523  2        PAT$FREERELEASE(.PAT$GL_TEMP_BUF[DSC$A_POINTER], (.PAT$GL_TEMP_BUF[DSC$W_LENGTH]+3)/4);
;  870          3524  2        PAT$GL_TEMP_BUF[DSC$W_LENGTH] = 0;
;  871          3525  2        PAT$GL_TEMP_BUF[DSC$A_POINTER] = 0;
;  872          3526  2        !++
;  873          3527  2        ! If this was a VERIFY command, we are all done.  Return for next command.
;  874          3528  2        !--
;  875          3529  2        IF .PAT$GL_CONTEXT[VERIFY_BIT]
;  876          3530  2        THEN
;  877          3531  2                RETURN;
;  878          3532  2
;  879          3533  2        !++
;  880          3534  2        ! Check if old instruction should be moved to patch area, i.e., is this an
;  881          3535  2        ! INSERT command.  Remember the number of bytes of old instructions moved
;  882          3536  2        ! in case there are forward referenced symbols to relocate in the new
;  883          3537  2        ! instructions.
;  884          3538  2        !--
;  885          3539  2        IF .PAT$GL_CONTEXT [INSERT_BIT]
;  886          3540  2        THEN
;  887          3541  3                BEGIN
;  888          3542  3                PAT$FILL_BUF(PAT$GL_TEMP_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]);
```

```
;   889          3543  3          OLD_INS_SIZ = .PAT$GL_TEMP_BUF[DSC$W_LENGTH];                   ! Remember # of bytes of old instructions mo
;   890          3544  3          END
;   891          3545  2     ELSE
;   892          3546  2          BEGIN
;   893          3547  2          OLD_INS_SIZ = 0;                                                ! No old instructions moved
;   894          3548  2          NEXT_LOC = .UNMAPPED_LOC;                                       ! Set next deposit location for REPLACE comm
;   895          3549  2          END;
;   896          3550  2
;   897          3551  2     !++
;   898          3552  2     ! Now fit the replacement value/instruction into the location.
;   899          3553  2     !--
;   900          3554  2     IF (NEW_INS_PTR = .LIST_ELEM_FLINK(.POINTER)) EQLA 0                    ! If no replacement argument
;   901          3555  2     THEN                                                                    ! then report error
;   902          3556  2          SIGNAL(PAT$_INVCMD);
;   903          3557  2     PAT$GL_SYMTBPTR = .PAT$GL_NEWLABLS;                                     ! Use the new contents label table
;   904          3558  2
;   905          3559  2     !++
;   906          3560  2     ! Now build a buffer containing the new values to be deposited.  The deposits
;   907          3561  2     ! are not done directly to memory in case part of the command is invalid.
;   908          3562  2     !--
;   909          3563  2     WHILE (POINTER = .LIST_ELEM_FLINK(.POINTER)) NEQA 0                     ! Point to next argument
;   910          3564  2     DO
;   911          3565  3          BEGIN
;   912          3566  3          IF .PAT$GB_MOD_PTR [MODE_INSTRUC]                               ! Test for instruction or data replacement
;   913          3567  3          THEN
;   914          3568  4               BEGIN
;   915          3569  4               !++
;   916          3570  4               ! Now encode the replacement instruction.
;   917          3571  4               !--
;   918          3572  4               FILL_CHAR = NOP_INSTR;                                  ! Set the fill character
;   919          3573  4               IF NOT PAT$INS_ENCODE(.LIST_ELEM_EXP1(.POINTER), INSTRUC_BUF,
;   920          3574  4                                    .NEXT_LOC, PAT$GL_NEW_ASD, PAT$GL_TEMP_BUF)
;   921          3575  4               THEN
;   922          3576  4                    SIGNAL(PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER));
;   923          3577  4               PAT$FILL_BUF(PAT$GL_TEMP_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]); ! Insert instruction into te
;   924          3578  4               NEXT_LOC = .NEXT_LOC + .INSTRUC_BUF[0];
;   925          3579  4               END
;   926          3580  3          ELSE
;   927          3581  3               !++
;   928          3582  3               ! The replacement is for a value.  Therefore it is on the parse
;   929          3583  3               ! stack.  Check for a truncation error.  Then set the fill
;   930          3584  3               ! character and write the value to the temporary buffer.
;   931          3585  3               !--
;   932          3586  4               BEGIN
;   933          3587  4               IF .LIST_ELEM_EXP1(.POINTER) LSS 0
;   934          3588  4               THEN
;   935          3589  5                    BEGIN
;   936          3590  5                    IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
;   937          3591  5                    THEN
;   938          3592  5                         SIGNAL(PAT$_NUMTRUNC);
;   939          3593  5                    END
;   940          3594  4               ELSE
;   941          3595  4                    IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
;   942          3596  4                    THEN
;   943          3597  4                         SIGNAL(PAT$_NUMTRUNC);
;   944          3598  4               FILL_CHAR = ZERO_BYTE;                                  ! Set the fill character
;   945          3599  4               PAT$FILL_BUF(PAT$GL_TEMP_BUF, LIST_ELEM_EXP1(.POINTER), .VAL_SIZ);
```

```
;  946     3600  4                   NEXT_LOC = .NEXT_LOC + .VAL_SIZ;
;  947     3601  4                   END;
;  948     3602              END;
;  949     3603  !++
;  950     3604  ! Resolve the forward references in symbolic instruction operands.
;  951     3605  !--
;  952     3606  PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
;  953     3607
;  954     3608  !++
;  955     3609  ! Now check the replacement size against old instruction size.
;  956     3610  !--
;  957     3611  IF .PAT$GL_TEMP_BUF[DSC$W_LENGTH] LSS .HOLE_SIZ                ! Make temporary buffer at least as large as
;  958     3612  THEN
;  959     3613          BEGIN
;  960     3614          LOCAL
;  961     3615                  TEMP_PTR;                                       ! Temporary pointer to temporary buffer
;  962     3616
;  963     3617          TEMP_PTR = PAT$FREEZ((.HOLE_SIZ + A_LONGWORD - 1)/A_LONGWORD);
;  964     3618          CH$COPY(.PAT$GL_TEMP_BUF[DSC$W_LENGTH], .PAT$GL_TEMP_BUF[DSC$A_POINTER],
;  965     3619                  .FILL_CHAR, .HOLE_SIZ, .TEMP_PTR);
;  966     3620          PAT$FREERELEASE(.PAT$GL_TEMP_BUF[DSC$A_POINTER], (.PAT$GL_TEMP_BUF[DSC$W_LENGTH] +3)/4);
;  967     3621          PAT$GL_TEMP_BUF[DSC$A_POINTER] = CH$PTR(.TEMP_PTR, 0);
;  968     3622          PAT$GL_TEMP_BUF[DSC$W_LENGTH] = .HOLE_SIZ;
;  969     3623          END;
;  970     3624
;  971     3625  !++
;  972     3626  ! Now write the temporary buffer over the mapped input image.
;  973     3627  !--
;  974     3628  IF .PAT$GL_TEMP_BUF[DSC$W_LENGTH] EQL .HOLE_SIZ
;  975     3629  THEN
;  976     3630          BEGIN
;  977     3631          !++
;  978     3632          ! Replacement data fits.  Write it to memory and output new contents.
;  979     3633          !--
;  980     3634          PAT$WRITE_MEM(.UNMAPPED_LOC, .PAT$GL_TEMP_BUF[DSC$A_POINTER], .PAT$GL_TEMP_BUF[DSC$W_LENGTH]);
;  981     3635          NEXT_LOC = .UNMAPPED_LOC + .HOLE_SIZ;
;  982     3636          PAT$GL_NEXT_LOC = .UNMAPPED_LOC;
;  983     3637          WHILE .PAT$GL_NEXT_LOC LSSA .NEXT_LOC                   ! Output new contents
;  984     3638          DO
;  985     3639                  PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, CASE_TABLE);
;  986     3640          END
;  987     3641  ELSE
;  988     3642          BEGIN
;  989     3643          !++
;  990     3644          ! The replacement instruction is too large.  It
;  991     3645          ! must be relocated to the patch area.
;  992     3646          !--
;  993     3647          IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
;  994     3648          THEN
;  995     3649                  RELOCAT_INS(.UNMAPPED_LOC, .HOLE_SIZ, .OLD_INS_SIZ, .NEW_INS_PTR)
;  996     3650          ELSE
;  997     3651                  SIGNAL(PAT$_REPLACEERR);                        ! Internal error if patch area needed for da
;  998     3652          END;
;  999     3653  !++
; 1000     3654  ! Now add all the new labels to the user-defined symbol table.
; 1001     3655  !--
; 1002     3656  PAT$ADD_LABELS(PAT$GL_OLDLABLS);
```

```
;  1003        3657  2 PAT$ADD_LABELS(PAT$GL_NEWLABLS);
;  1004        3658  2 PAT$ADD_LABELS(PAT$GL_RLCLABLS);
;  1005        3659  2 RETURN;
;  1006        3660  1 END;                                                    ! End of PAT$REPLACE_CMD


                                                  0FFC 00000          .ENTRY  PAT$REPLACE_CMD, Save R2,R3,R4,R5,R6,R7,R8,-      ; 3315
                                                                              R9,R10,R11
                                       5E      FF2C   CE   9E 00002           MOVAB   -212(SP), SP
                                       57 00000000G   EF   D0 00007           MOVL    PAT$GL_HEAD_LST, POINTER                  ; 3415
                                                      04   13 0000E           BEQL    1$                                        ; 3416
                                                      67   D5 00010           TSTL    (POINTER)
                                                      0D   12 00012           BNEQ    2$
                                          006D80DA    8F   DD 00014 1$:       PUSHL   #7176410                                  ; 3418
                                 00000000G  00        01   FB 0001A           CALLS   #1, LIB$SIGNAL
                                 00000000G  EF        01   88 00021 2$:       BISB2   #1, PAT$GL_CONTEXT+1                      ; 3423
                                       5B       04   A7   D0 00028           MOVL    4(POINTER), UNMAPPED_LOC                  ; 3424
                                       59       04   A7   D0 0002C           MOVL    4(POINTER), NEXT_LOC                      ; 3425
                                                      56   D4 00030           CLRL    HOLE_SIZ                                  ; 3426
                                 00000000G  EF 00000000G   EF   D0 00032     MOVL    PAT$GL_OLDLABLS, PAT$GL_SYMTBPTR          ; 3427
                                       57               67   D0 0003D 3$:    MOVL    (POINTER), POINTER                        ; 3435
                                       52       04   A7   9E 00040           MOVAB   4(POINTER), R2                            ; 3451
                                       50 00000000G   EF   D0 00044           MOVL    PAT$GB_MOD_PTR, R0                        ; 3441
                                       38       03   A0   E9 0004B           BLBC    3(R0), 5$
                                          00000000G   EF   9F 0004F           PUSHAB  PAT$GL_TEMP_BUF                           ; 3451
                                          00000000G   EF   9F 00055           PUSHAB  PAT$GL_OLD_ASD
                                                      59   DD 0005B           PUSHL   NEXT_LOC                                  ; 3452
                                                B0    AD   9F 0005D           PUSHAB  INSTRUC_BUF                               ; 3451
                                                      62   DD 00060           PUSHL   (R2)
                                 00000000G  EF              05   FB 00062     CALLS   #5, PAT$INS_ENCODE
                                                11        50   E8 00069     BLBS    R0, 4$
                                                      62   DD 0006C           PUSHL   (R2)                                      ; 3454
                                                      01   DD 0006E           PUSHL   #1
                                          006D810A    8F   DD 00070           PUSHL   #7176458
                                 00000000G  00        03   FB 00076           CALLS   #3, LIB$SIGNAL
                                       53       B1   AD   9E 0007D 4$:       MOVAB   INSTRUC_BUF+1, OLD_VALUE_PTR              ; 3455
                                       54       B0   AD   9A 00081           MOVZBL  INSTRUC_BUF, VAL_SIZ                      ; 3456
                                                      2D   11 00085           BRB     8$                                        ; 3441
                                       53               52   D0 00087 5$:    MOVL    R2, OLD_VALUE_PTR                         ; 3466
                                       54       01   A0   9A 0008A           MOVZBL  1(R0), VAL_SIZ                            ; 3467
                                 50    54               03   78 0008E           ASHL    #3, VAL_SIZ, R0                           ; 3471
                                                      62   D5 00092           TSTL    (R2)                                      ; 3468
                                                      07   18 00094           BGEQ    6$
                           51    62    50               00   EE 00096           EXTV    #0, R0, (R2), R1                          ; 3471
                                                      05   11 0009B           BRB     7$
                           51    62    50               00   EF 0009D 6$:       EXTZV   #0, R0, (R2), R1                          ; 3476
                                       62               51   D1 000A2 7$:       CMPL    R1, (R2)
                                                      0D   13 000A5           BEQL    8$
                                          006D8023    8F   DD 000A7           PUSHL   #7176227                                  ; 3478
                                 00000000G  00        01   FB 000AD           CALLS   #1, LIB$SIGNAL
                                                      18   BB 000B4 8$:       PUSHR   #^M<R3,R4>                                ; 3481
                                          00000000G   EF   9F 000B6           PUSHAB  PAT$GL_TEMP_BUF
                                 00000000V  EF        03   FB 000BC           CALLS   #3, PAT$FILL_BUF
                                       56               54   C0 000C3           ADDL2   VAL_SIZ, HOLE_SIZ                         ; 3482
```

```
                                7E  D4 000C6          CLRL    -(SP)                                                   ; 3484
                    00000000G   EF  9F 000C8          PUSHAB  PAT$GL_OLD_ASD
                    00000000'   EF  9F 000CE          PUSHAB  OLD_TAB_STG
                                59  DD 000D4          PUSHL   NEXT_LOC
    00000000V   EF              04  FB 000D6          CALLS   #4, PAT$OUT_MEM_LOC
                59  00000000G   EF  D0 000DD          MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                               ; 3485
                0A        08    A7  D1 000E4          CMPL    8(POINTER), #10                                         ; 3490
                                03  13 000E8          BEQL    9$
                              FF50  31 000EA          BRW     3$
                    00000000G   EF  9F 000ED 9$:      PUSHAB  PAT$GL_TEMP_BUF                                         ; 3498
    00000000G   EF              01  FB 000F3          CALLS   #1, PAT$RESOLVE_INS
                                55  D4 000FA          CLRL    BYTE_COUNT                                              ; 3504
                56              55  D1 000FC 10$:     CMPL    BYTE_COUNT, HOLE_SIZ                                    ; 3505
                                44  18 000FF          BGEQ    13$
58              56              55  C3 00101          SUBL3   BYTE_COUNT, HOLE_SIZ, BUF_SIZE                          ; 3508
    00000084    8F              58  D1 00105          CMPL    BUF_SIZE, #132
                                04  15 0010C          BLEQ    11$
                58        84    8F  9A 0010E          MOVZBL  #132, BUF_SIZE                                          ; 3510
                      4100      8F  BB 00112 11$:     PUSHR   #^M<R8,SP>                                              ; 3512
                50  00000000G   EF  D0 00116          MOVL    PAT$GL_HEAD_LST, R0                                     ; 3511
                          04 B045   9F  9F 0011D          PUSHAB  @4(R0)[BYTE_COUNT]
    00000000G   EF              03  FB 00121          CALLS   #3, PAT$GET_VALUE
6E  00000000GFF45               58  29 00128          CMPC3   BUF_SIZE, @PAT$GL_TEMP_BUF+4[BYTE_COUNT], -            ; 3513
                                                              OLD_CONTENTS
                                0D  13 00131          BEQL    12$                                                     ; 3516
                    006D8290    8F  DD 00133          PUSHL   #7176848
    00000000G   00              01  FB 00139          CALLS   #1, LIB$SIGNAL
                55              58  C0 00140 12$:     ADDL2   BUF_SIZE, BYTE_COUNT                                    ; 3517
                                B7  11 00143          BRB     10$                                                     ; 3505
                50  00000000G   EF  3C 00145 13$:     MOVZWL  PAT$GL_TEMP_BUF, R0                                     ; 3523
                50              03  C0 0014C          ADDL2   #3, R0
7E              50              04  C7 0014F          DIVL3   #4, R0, -(SP)
                    00000000G   EF  DD 00153          PUSHL   PAT$GL_TEMP_BUF+4
    00000000G   EF              02  FB 00159          CALLS   #2, PAT$FREERELEASE
                    00000000G   EF  B4 00160          CLRW    PAT$GL_TEMP_BUF                                         ; 3524
                    00000000G   EF  D4 00166          CLRL    PAT$GL_TEMP_BUF+4                                       ; 3525
01  00000000G   EF              05  E1 0016C          BBC     #5, PAT$GL_CONTEXT+2, 14$                               ; 3529
                                04  00174          RET
                    00000000G   EF  95 00175 14$:     TSTB    PAT$GL_CONTEXT+2                                        ; 3539
                                1D  18 0017B          BGEQ    15$
                7E        B0    AD  9A 0017D          MOVZBL  INSTRUC_BUF, -(SP)                                      ; 3542
                          B1    AD  9F 00181          PUSHAB  INSTRUC_BUF+1
                    00000000G   EF  9F 00184          PUSHAB  PAT$GL_TEMP_BUF
    00000000V   EF              03  FB 0018A          CALLS   #3, PAT$FILL_BUF
                5A  00000000G   EF  3C 00191          MOVZWL  PAT$GL_TEMP_BUF, OLD_INS_SIZ                            ; 3543
                                05  11 00198          BRB     16$                                                     ; 3539
                                5A  D4 0019A 15$:     CLRL    OLD_INS_SIZ                                             ; 3547
                59              5B  D0 0019C          MOVL    UNMAPPED_LOC, NEXT_LOC                                  ; 3548
                58              67  D0 0019F 16$:     MOVL    (POINTER), NEW_INS_PTR                                  ; 3554
                                0D  12 001A2          BNEQ    17$                                                     ; 3556
                    006D80DA    8F  DD 001A4          PUSHL   #7176410
    00000000G   00              01  FB 001AA          CALLS   #1, LIB$SIGNAL
    00000000G   EF  00000000G   EF  D0 001B1 17$:     MOVL    PAT$GL_NEWLABLS, PAT$GL_SYMTBPTR                        ; 3557
                57              67  D0 001BC 18$:     MOVL    (POINTER), POINTER                                      ; 3563
                                03  12 001BF          BNEQ    19$
                              0099  31 001C1          BRW     26$
                52        04    A7  9E 001C4 19$:     MOVAB   4(POINTER), R2                                          ; 3573
```

```
                        50 00000000G   EF   D0 001C8          MOVL    PAT$GB_MOD_PTR, R0                              ; 3566
                        4E       03    A0   E9 001CF          BLBC    3(R0), 22$
                        53             01   90 001D3          MOVB    #1, FILL_CHAR                                   ; 3572
                                00000000G EF 9F 001D6          PUSHAB  PAT$GL_TEMP_BUF                                 ; 3573
                                00000000G EF 9F 001DC          PUSHAB  PAT$GL_NEW_ASD
                                          59 DD 001E2          PUSHL   NEXT_LOC                                        ; 3574
                                       B0   AD 9F 001E4          PUSHAB  INSTRUC_BUF                                     ; 3573
                                          62 DD 001E7          PUSHL   (R2)
                00000000G EF                05 FB 001E9          CALLS   #5, PAT$INS_ENCODE
                        11                  50 E8 001F0          BLBS    R0, 20$
                                            62 DD 001F3          PUSHL   (R2)                                            ; 3576
                                            01 DD 001F5          PUSHL   #1
                                    006D810A 8F DD 001F7         PUSHL   #7176458
                00000000G 00                03 FB 001FD          CALLS   #3, LIB$SIGNAL
                        7E       B0   AD 9A 00204 20$:          MOVZBL  INSTRUC_BUF, -(SP)                              ; 3577
                                 B1   AD 9F 00208               PUSHAB  INSTRUC_BUF+1
                                00000000G EF 9F 0020B           PUSHAB  PAT$GL_TEMP_BUF
                00000000V EF                03 FB 00211          CALLS   #3, PAT$FILL_BUF
                        50       B0   AD 9A 00218               MOVZBL  INSTRUC_BUF, R0                                 ; 3578
                        59            50 C0 0021C               ADDL2   R0, NEXT_LOC
                                      9B 11 0021F 21$:          BRB     18$                                             ; 3566
                50      54            03 78 00221 22$:          ASHL    #3, VAL_SIZ, R0                                 ; 3590
                                      62 D5 00225               TSTL    (R2)                                            ; 3587
                                      07 18 00227               BGEQ    23$
        51      62      50            00 EE 00229               EXTV    #0, R0, (R2), R1                                ; 3590
                                      05 11 0022E               BRB     24$
        51      62      50            00 EF 00230 23$:          EXTZV   #0, R0, (R2), R1                                ; 3595
                        62            51 D1 00235 24$:          CMPL    R1, (R2)
                                      0D 13 00238               BEQL    25$
                                006D8023 8F DD 0023A            PUSHL   #7176227                                        ; 3597
                00000000G 00          01 FB 00240               CALLS   #1, LIB$SIGNAL
                                      53 94 00247 25$:          CLRB    FILL_CHAR                                       ; 3598
                                      14 BB 00249               PUSHR   #^M<R2,R4>                                      ; 3599
                                00000000G EF 9F 0024B           PUSHAB  PAT$GL_TEMP_BUF
                00000000V EF                03 FB 00251          CALLS   #3, PAT$FILL_BUF
                        59            54 C0 00258               ADDL2   VAL_SIZ, NEXT_LOC                               ; 3600
                                      C2 11 0025B               BRB     21$                                             ; 3563
                                00000000G EF 9F 0025D 26$:      PUSHAB  PAT$GL_TEMP_BUF                                 ; 3606
                00000000G EF                01 FB 00263          CALLS   #1, PAT$RESOLVE_INS
56 00000000G    EF      10            00 ED 0026A               CMPZV   #0, #16, PAT$GL_TEMP_BUF, HOLE_SIZ              ; 3611
                                      49 18 00273               BGEQ    27$
                        50       03   A6 9F 00275               MOVAB   3(R6), R0                                       ; 3617
                7E      50            04 C7 00279               DIVL3   #4, R0, -(SP)
                00000000G EF          01 FB 0027D               CALLS   #1, PAT$FREEZ
                        57            50 D0 00284               MOVL    R0, TEMP_PTR
56              53 00000000G FF 00000000G EF 2C 00287           MOVC5   PAT$GL_TEMP_BUF, @PAT$GL_TEMP_BUF+4, -          ; 3619
                                          67 00294                      FILL_CHAR, HOLE_SIZ, (TEMP_PTR)
                        50 00000000G EF   3C 00295              MOVZWL  PAT$GL_TEMP_BUF, R0                             ; 3620
                        50            03 C0 0029C               ADDL2   #3, R0
                7E      50            04 C7 0029F               DIVL3   #4, R0, -(SP)
                                00000000G EF DD 002A3           PUSHL   PAT$GL_TEMP_BUF+4
                00000000G EF                02 FB 002A9          CALLS   #2, PAT$FREERELEASE
                00000000G EF                57 D0 002B0          MOVL    TEMP_PTR, PAT$GL_TEMP_BUF+4                     ; 3621
                00000000G EF                56 B0 002B7          MOVW    HOLE_SIZ, PAT$GL_TEMP_BUF                       ; 3622
56 00000000G    EF      10            00 ED 002BE 27$:          CMPZV   #0, #16, PAT$GL_TEMP_BUF, HOLE_SIZ              ; 3628
                                      47 12 002C7               BNEQ    29$
                        7E 00000000G EF   3C 002C9              MOVZWL  PAT$GL_TEMP_BUF, -(SP)                          ; 3634
```

```
                        00000000G   EF   DD 002D0          PUSHL   PAT$GL_TEMP_BUF+4                        ;
                                    5B   DD 002D6          PUSHL   UNMAPPED_LOC                             ;
                00000000G   EF          03   FB 002D8      CALLS   #3, PAT$WRITE_MEM                        ;
             59 5B                      56   C1 002DF      ADDL3   HOLE_SIZ, UNMAPPED_LOC, NEXT_LOC        ; 3635
                00000000G   EF          5B   D0 002E3      MOVL    UNMAPPED_LOC, PAT$GL_NEXT_LOC           ; 3636
                            59 00000000G EF  D1 002EA 28$:  CMPL    PAT$GL_NEXT_LOC, NEXT_LOC               ; 3637
                                        46   1E 002F1      BGEQU   31$                                     ;
                                        01   DD 002F3      PUSHL   #1                                      ; 3639
                            00000000G   EF   9F 002F5      PUSHAB  PAT$GL_NEW_ASD                          ;
                            00000000'   EF   9F 002FB      PUSHAB  NEW_TAB_STG                             ;
                            00000000G   EF   DD 00301      PUSHL   PAT$GL_NEXT_LOC                         ;
                00000000V   EF              04   FB 00307  CALLS   #4, PAT$OUT_MEM_LOC                     ;
                                        DA   11 0030E      BRB     28$                                     ;
                            50 00000000G EF  D0 00310 29$:  MOVL    PAT$GB_MOD_PTR, R0                      ; 3647
                            11      03       A0   E9 00317  BLBC    3(R0), 30$                              ;
                                        58   DD 0031B      PUSHL   NEW_INS_PTR                             ; 3649
                                  0440  8F   BB 0031D      PUSHR   #^M<R6,R10>                             ;
                                        5B   DD 00321      PUSHL   UNMAPPED_LOC                            ;
                00000000V   EF              04   FB 00323  CALLS   #4, RELOCAT_INS                         ;
                                        0D   11 0032A      BRB     31$                                     ;
                        006D815A    8F   DD 0032C 30$:    PUSHL   #7176538                                ; 3651
                00000000G   00          01   FB 00332      CALLS   #1, LIB$SIGNAL                          ;
                            00000000G   EF   9F 00339 31$:  PUSHAB  PAT$GL_OLDLABLS                         ; 3656
                00000000G   EF          01   FB 0033F      CALLS   #1, PAT$ADD_LABELS                      ;
                            00000000G   EF   9F 00346      PUSHAB  PAT$GL_NEWLABLS                         ; 3657
                00000000G   EF          01   FB 0034C      CALLS   #1, PAT$ADD_LABELS                      ;
                            00000000G   EF   9F 00353      PUSHAB  PAT$GL_RLCLABLS                         ; 3658
                00000000G   EF          01   FB 00359      CALLS   #1, PAT$ADD_LABELS                      ;
                                        04 00360           RET                                             ; 3660

; Routine Size:  865 bytes,    Routine Base:  _PAT$CODE + 030A
```

```
;  1008     3661  1 ROUTINE RELOCAT_INS (OLD_LOC, HOLE_SIZE, OLD_INS_SIZ, ASC_INS_PTR) : NOVALUE =
;  1009     3662  1
;  1010     3663  1 !++
;  1011     3664  1 ! FUNCTIONAL DESCRIPTION:
;  1012     3665  1 !
;  1013     3666  1 !       This routine relocates an instruction from an old address to the patch
;  1014     3667  1 !       area.  It then moves in any new instructions, specified as an argument
;  1015     3668  1 !       list for a patch command.  A branch or jump instruction is then put
;  1016     3669  1 !       into the old address.  If there is not enough room left by the
;  1017     3670  1 !       removal of the instruction, then more instructions are moved to the
;  1018     3671  1 !       patch area until the branch instruction iwll fit.  Lastly, a return
;  1019     3672  1 !       branch instruction is placed in the patch area to return execution
;  1020     3673  1 !       to the next sequential instruction past the old address.
;  1021     3674  1 !
;  1022     3675  1 !       Any new instructions to be inserted are in a command argument
;  1023     3676  1 !       list, created by the parser.  Each argument entry is made up of
;  1024     3677  1 !       three longwords. The first is a forward link to the next entry.
;  1025     3678  1 !       The second longword contains the address of a counted byte stream
;  1026     3679  1 !       that is to be translated into a binary instruction which is
;  1027     3680  1 !       to be inserted into the patch area.  The third longword is unused.
;  1028     3681  1 !
;  1029     3682  1 ! CALLING SEQUENCE:
;  1030     3683  1 !
;  1031     3684  1 !       RELOCATE_CMD (OLD_LOCATION, NEW_INSTRUCTION_PTR)
;  1032     3685  1 !
;  1033     3686  1 ! INPUTS:
;  1034     3687  1 !
;  1035     3688  1 !       OLD_LOC - Unmapped address of instruction to be moved
;  1036     3689  1 !       HOLE_SIZE - Number of free bytes at OLD_LOC
;  1037     3690  1 !       OLD_INS_SIZ - Number of bytes of old instruction preceding new instruction
;  1038     3691  1 !       ASC_INS_PTR - Pointer to first new instruction on command argument list
;  1039     3692  1 !
;  1040     3693  1 ! IMPLICIT INPUTS:
;  1041     3694  1 !
;  1042     3695  1 !       PAT$GL_TEMP_BUF - String descriptor for counted binary instructions
;  1043     3696  1 !
;  1044     3697  1 !       The head of the linked list, the current mode, and
;  1045     3698  1 !       the current patch area descriptor.
;  1046     3699  1 !
;  1047     3700  1 ! OUTPUTS:
;  1048     3701  1 !
;  1049     3702  1 !       none
;  1050     3703  1 !
;  1051     3704  1 ! IMPLICIT OUTPUTS:
;  1052     3705  1 !
;  1053     3706  1 !       NONE
;  1054     3707  1 !
;  1055     3708  1 ! ROUTINE VALUE:
;  1056     3709  1 !
;  1057     3710  1 !       novalue
;  1058     3711  1 !
;  1059     3712  1 ! SIDE EFFECTS:
;  1060     3713  1 !
;  1061     3714  1 !       If the default patch area is to be used and it does not currently
;  1062     3715  1 !       exist when PAT$EXP_AREA is called PAT$BUILD_ISE is invoked which
;  1063     3716  1 !       given the address of the instructions to be moved will propagate
;  1064     3717  1 !       the image section attributes of the old image section to the newly
```

```
;  1065     3718  1 !       created default patch area image section descriptor.
;  1066     3719  1 !
;  1067     3720  1 !       The patch area now contains the moved instruction and the new ones
;  1068     3721  1 !       plus a branch instruction back to the inline code.  The old
;  1069     3722  1 !       instruction location contains a branch to the patch area.
;  1070     3723  1 !       If a failure in a write or contents verification occurs,
;  1071     3724  1 !       the routine returns immediately.
;  1072     3725  1 !
;  1073     3726  1 !--
;  1074     3727  1
;  1075     3728  2 BEGIN
;  1076     3729  2
;  1077     3730  2 LITERAL
;  1078     3731  2         MAX_BYTE_DISP = 127,                    ! Maximum displacement for BRB
;  1079     3732  2         MIN_BYTE_DISP = -128,                   ! Minimum displacement for BRB
;  1080     3733  2         MAX_WORD_DISP = 32767,                  ! Maximum displacement for BRW
;  1081     3734  2         MIN_WORD_DISP = -32768,                 ! Minimum displacement for BRW
;  1082     3735  2         BRB_OPCODE = %X'11',                    ! Opcode for BRB
;  1083     3736  2         BRW_OPCODE = %X'31',                    ! Opcode for BRW
;  1084     3737  2         JMP_OPCODE = %X'17',                    ! Opcode for JMP
;  1085     3738  2         BRB_INS_SIZ = 2,                        ! Size of BRB instruction
;  1086     3739  2         BRW_INS_SIZ = 3,                        ! Size of BRW instruction
;  1087     3740  2         JMP_INS_SIZ = 6,                        ! Size of JMP instruction
;  1088     3741  2         PC_DEFERRED = %X'EF',                   ! PC deferred instruction mode
;  1089     3742  2         NOP_INSTR = 1,                          ! Fill character for instruction
;  1090     3743  2         MAX_INST_LEN    = 80;                   ! Maximum number of binary bytes in an instr
;  1091     3744  2
;  1092     3745  2 LOCAL
;  1093     3746  2         SUCC_OLD_INS,                           ! Number of bytes of successive old instruct
;  1094     3747  2         DECODED_INS,                            ! Pointer to ascii instruction
;  1095     3748  2         NXT_ASC_INS,                            ! Pointer to next ascii instruction in argum
;  1096     3749  2         NEXT_PC,                                ! PC of next instruction to decode
;  1097     3750  2         NEW_INS_PTR : REF VECTOR[,BYTE],        ! Pointer to relocated instruction stream
;  1098     3751  2         BR_DISPLACEMENT : SIGNED LONG,          ! Displacement for branch instruction
;  1099     3752  2         BR_INSTRUC : VECTOR[JMP_INS_SIZ+1,BYTE],! Encoded counted string branch instruction
;  1100     3753  2         NEW_LOC,                                ! Address in patch area for relocated instru
;  1101     3754  2         NEXT_LOC,                               ! Address of next instruction of inline code
;  1102     3755  2         CUR_LOC,                                ! Address of current instruction to be moved
;  1103     3756  2         LOCAL_BUF : VECTOR[MAX_INST_LEN, BYTE], ! Local buffer for binary instruction stream
;  1104     3757  2         INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE]; ! Local buffer for ascic instructions
;  1105     3758  2
;  1106     3759  2 !++
;  1107     3760  2 ! Enable instruction substitution.
;  1108     3761  2 !--
;  1109     3762  2 PAT$GL_CONTEXT[INST_SUBST] = TRUE;
;  1110     3763  2 PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;
;  1111     3764  2
;  1112     3765  2 !++
;  1113     3766  2 ! Check that there is enough room in the patch area for the instructions
;  1114     3767  2 ! encoded in the temporary buffer, PAT$GL_TEMP_BUF.  This is the minimum size
;  1115     3768  2 ! that may be required.  Instruction substitution may enlarge this size.  This
;  1116     3769  2 ! will also insure that a patch area address is defined.
;  1117     3770  2 !--
;  1118     3771  3 IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
;  1119     3772  2 THEN
;  1120     3773  3         BEGIN
;  1121     3774  4         IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
```

```
;  1122          3775  3          THEN
;  1123          3776  4                  BEGIN
;  1124          3777  4                  PAT$EXP_AREA((.PAT$GL_TEMP_BUF[DSC$W_LENGTH] + A_PAGE - 1)/A_PAGE, ,OLD_LOC);
;  1125          3778  5                  IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
;  1126          3779  4                  THEN
;  1127          3780  4                          SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
;  1128          3781  4                           .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  1129          3782  4                  END
;  1130          3783  3          ELSE
;  1131          3784  3                  SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
;  1132          3785  3                   .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  1133          3786  3          END;
;  1134          3787
;  1135          3788  2  !++
;  1136          3789  2  ! Set pointer to relocation address.
;  1137          3790  2  !--
;  1138          3791  2  NEW_LOC = CH$PTR(.PAT$GL_PATAREA[DSC$A_POINTER], 0);
;  1139          3792  2
;  1140          3793  2  !++
;  1141          3794  2  ! Now compute the branch displacement size.  Then build the binary code
;  1142          3795  2  ! based on the displacement.
;  1143          3796  2  !--
;  1144          3797  2  BR_DISPLACEMENT = .NEW_LOC - .OLD_LOC - BRB_INS_SIZ;
;  1145          3798  2  IF (.BR_DISPLACEMENT LEQ MAX_BYTE_DISP) AND (.BR_DISPLACEMENT GEQ MIN_BYTE_DISP)
;  1146          3799  2  THEN
;  1147          3800  2          BEGIN
;  1148          3801  2          BR_INSTRUC[0] = BRB_INS_SIZ;
;  1149          3802  2          BR_INSTRUC[1] = BRB_OPCODE;
;  1150          3803  2          CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[2],0));
;  1151          3804  2          END
;  1152          3805  2  ELSE
;  1153          3806  2          IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)
;  1154          3807  2          THEN
;  1155          3808  2                  BEGIN
;  1156          3809  2                  BR_INSTRUC[0] = BRW_INS_SIZ;
;  1157          3810  2                  BR_INSTRUC[1] = BRW_OPCODE;
;  1158          3811  2                  BR_DISPLACEMENT = .BR_DISPLACEMENT - (BRW_INS_SIZ - BRB_INS_SIZ);
;  1159          3812  2                  CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[2],0));
;  1160          3813  2                  END
;  1161          3814  2          ELSE
;  1162          3815  2                  BEGIN
;  1163          3816  2                  BR_INSTRUC[0] = JMP_INS_SIZ;
;  1164          3817  2                  BR_INSTRUC[1] = JMP_OPCODE;
;  1165          3818  2                  BR_INSTRUC[2] = PC_DEFERRED;
;  1166          3819  2                  BR_DISPLACEMENT = .BR_DISPLACEMENT - (JMP_INS_SIZ - BRB_INS_SIZ);
;  1167          3820  2                  CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[3],0));
;  1168          3821  2                  END;
;  1169          3822  2
;  1170          3823  2  !++
;  1171          3824  2  ! Now see if the branch instruction will fit in the hole left at the old
;  1172          3825  2  ! location.  If not, then move more instructions to the patch area until it
;  1173          3826  2  ! will fit.
;  1174          3827  2  !--
;  1175          3828  2  NEXT_LOC = .OLD_LOC + .HOLE_SIZE;                          ! Compute address of next inline instruction
;  1176          3829  2  NEXT_PC = .OLD_LOC + .HOLE_SIZE;                          ! Compute address of next inline instruction
;  1177          3830  2  SUCC_OLD_INS = .PAT$GL_TEMP_BUF[DSC$W_LENGTH];            ! Remember where extra old instructions move
;  1178          3831  2  WHILE .BR_INSTRUC[0] GTR .HOLE_SIZE
```

```
;  1179          3832  2 DO
;  1180          3833                BEGIN
;  1181          3834                !++
;  1182          3835                ! First decode the instruction at the next old location.  Then
;  1183          3836                ! re-encode it to get the binary stream.  Then insert it into
;  1184          3837                ! the temporary buffer.
;  1185          3838                !--
;  1186          3839                PAT$OUT_MEM_LOC(.NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, NO_CASE_TABLE);
;  1187          3840                PAT$GL_BUF_SIZ = 0;
;  1188          3841                PAT$CP_OUT_STR = CH$PTR(LOCAL_BUF[1]);
;  1189          3842
;  1190          3843                !++
;  1191          3844                ! Because the instruction is actually in the code, the PC is the
;  1192          3845                ! same as the address of the byte stream.  PAT$INS_DECODE will update
;  1193          3846                ! the NEW_PC automatically.  To determine the length of the instruction
;  1194          3847                ! binary stream just decoded (for updating the HOLE_SIZE), NEXT_LOC must
;  1195          3848                ! be preserved.  Therefore, the return value from the routine is written
;  1196          3849                ! into NEW_PC not NEXT_LOC.  This will be the same value, unless the
;  1197          3850                ! routine failed.
;  1198          3851                !--
;  1199          3852                IF (NEXT_PC = PAT$INS_DECODE(.NEXT_LOC, 0, NEXT_PC, PAT$GL_OLD_ASD, NO_CASE_TABLE)) EQL 0
;  1200          3853                THEN
;  1201          3854                        SIGNAL(PAT$_NODECODE);
;  1202          3855                LOCAL_BUF[0] = .PAT$GL_BUF_SIZ;
;  1203          3856                NEW_INS_PTR = CH$PTR(INSTRUC_BUF, 0);
;  1204          3857                IF NOT PAT$INS_ENCODE(LOCAL_BUF, INSTRUC_BUF,
;  1205          3858                        .OLD_LOC + .PAT$GL_TEMP_BUF[DSC$W_LENGTH], PAT$GL_NEW_ASD, PAT$GL_TEMP_BUF)
;  1206          3859                THEN
;  1207          3860  4                     IF (.PAT$GB_SUBST_IN[0] NEQ 0)
;  1208          3861                        THEN
;  1209          3862                                NEW_INS_PTR = CH$PTR(PAT$GB_SUBST_IN, 0)
;  1210          3863                        ELSE
;  1211          3864                                SIGNAL(PAT$_NOENCODE, 1, LOCAL_BUF);
;  1212          3865                !++
;  1213          3866                ! There is a temporary restriction on relocation of CASE instructions
;  1214          3867                !--
;  1215          3868                IF (.NEW_INS_PTR[1] EQL OP_CASEB) OR
;  1216          3869                   (.NEW_INS_PTR[1] EQL OP_CASEW) OR
;  1217          3870  4                (.NEW_INS_PTR[1] EQL OP_CASEL)
;  1218          3871                THEN
;  1219          3872                        SIGNAL(PAT$_NORELOC + MSG$K_SEVERE);
;  1220          3873                PAT$FILL_BUF(PAT$GL_TEMP_BUF, NEW_INS_PTR[1], .NEW_INS_PTR[0]);
;  1221          3874                HOLE_SIZE = .HOLE_SIZE + .NEXT_PC - .NEXT_LOC;
;  1222          3875                NEXT_LOC = .NEXT_PC;
;  1223          3876                END;
;  1224          3877
;  1225          3878  !++
;  1226          3879  ! Decode the instructions in the temporary buffer and re-encode them at the
;  1227          3880  ! patch area address.  This will alter the addresses within the instructions.
;  1228          3881  !--
;  1229          3882  CUR_LOC = .PAT$GL_TEMP_BUF[DSC$A_POINTER];
;  1230          3883  NEXT_PC = .OLD_LOC;
;  1231          3884  NXT_ASC_INS = .ASC_INS_PTR;                                   ! Get pointer to next new instruction argume
;  1232          3885  WHILE .CUR_LOC LSSA (.PAT$GL_TEMP_BUF[DSC$A_POINTER] + .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
;  1233          3886  DO
;  1234          3887                BEGIN
;  1235          3888  3             PAT$GL_BUF_SIZ = 0;
```

```
;  1236     3889  3          PAT$CP_OUT_STR = CH$PTR(LOCAL_BUF[1]);
;  1237     3890  3          IF (.CUR_LOC GEQA .PAT$GL_TEMP_BUF[DSC$A_POINTER] + .OLD_INS_SIZ) AND
;  1238     3891  4              (.CUR_LOC LSSA .PAT$GL_TEMP_BUF[DSC$A_POINTER] + .SUCC_OLD_INS)
;  1239     3892  3          THEN
;  1240     3893  4                  BEGIN
;  1241     3894  4                  !++
;  1242     3895  4                  ! Take the new instructions out of the argument list in
;  1243     3896  4                  ! case there are any labels which will be relocated.
;  1244     3897  4                  ! This is only done for new instructions being deposited.
;  1245     3898  4                  ! The old instructions being relocated are decoded and re-encoded.
;  1246     3899  4                  !--
;  1247     3900  4                  DECODED_INS = .LIST_ELEM_EXP1(.NXT_ASC_INS);
;  1248     3901  4                  NXT_ASC_INS = .LIST_ELEM_FLINK(.NXT_ASC_INS);
;  1249     3902  4                  PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;                 ! Use reolcated label table
;  1250     3903  4                  END
;  1251     3904  3          ELSE
;  1252     3905  4                  BEGIN
;  1253     3906  4                  !++
;  1254     3907  4                  ! The instruction is an old instruction.  Therefore use the
;  1255     3908  4                  ! old label table and encode it from the decoded instruction.
;  1256     3909  4                  !--
;  1257     3910  4                  DECODED_INS = LOCAL_BUF;                            ! Point to ascii instruction
;  1258     3911  4                  PAT$GL_SYMTBPTR = .PAT$GL_OLDLABLS;                 ! Assume this is an old instruction
;  1259     3912  3                  END;
;  1260     3913  3          IF (CUR_LOC = PAT$INS_DECODE(.CUR_LOC, 0, NEXT_PC, PAT$GL_NEW_ASD, NO_CASE_TABLE)) EQL 0
;  1261     3914  3          THEN
;  1262     3915  3                  SIGNAL(PAT$_NODECODE);
;  1263     3916  3          LOCAL_BUF[0] = .PAT$GL_BUF_SIZ;
;  1264     3917  3          NEW_INS_PTR = CH$PTR(INSTRUC_BUF, 0);                       ! Set pointer to counted stream buffer
;  1265     3918  3          IF NOT PAT$INS_ENCODE(.DECODED_INS, INSTRUC_BUF,
;  1266     3919  3                    .NEW_LOC + .PAT$GL_RLOC_BUF[DSC$W_LENGTH], PAT$GL_NEW_ASD, PAT$GL_RLOC_BUF)
;  1267     3920  3          THEN
;  1268     3921  4                  IF (.PAT$GB_SUBST_IN[0] NEQ 0)
;  1269     3922  3                  THEN
;  1270     3923  3                          NEW_INS_PTR = CH$PTR(PAT$GB_SUBST_IN, 0)
;  1271     3924  3                  ELSE
;  1272     3925  3                          SIGNAL(PAT$_NOENCODE, 1, LOCAL_BUF);
;  1273     3926  3                  !++
;  1274     3927  3                  ! There is a temporary restriction on relocation of CASE instructions
;  1275     3928  3                  !--
;  1276     3929  3                  IF (.NEW_INS_PTR[1] EQL OP_CASEB) OR
;  1277     3930  3                     (.NEW_INS_PTR[1] EQL OP_CASEW) OR
;  1278     3931  4                     (.NEW_INS_PTR[1] EQL OP_CASEL)
;  1279     3932  3                  THEN
;  1280     3933  3                          SIGNAL(PAT$_NORELOC + MSG$K_SEVERE);
;  1281     3934  3                  PAT$FILL_BUF(PAT$GL_RLOC_BUF, NEW_INS_PTR[1], .NEW_INS_PTR[0]);
;  1282     3935  2                  END;
;  1283     3936  2          PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;                         ! Set relocated labels as default (old alway
;  1284     3937  2          PAT$RESOLVE_INS(PAT$GL_RLOC_BUF);
;  1285     3938  2
;  1286     3939  2          !++
;  1287     3940  2          ! Now a return branch instruction must be placed in the relocation buffer.
;  1288     3941  2          ! Compute the branch displacement size.  Then build the binary code based on
;  1289     3942  2          ! the displacement.
;  1290     3943  2          !--
;  1291     3944  3          BR_DISPLACEMENT = .NEXT_LOC - (.PAT$GL_PATAREA[DSC$A_POINTER] +
;  1292     3945  2                                  .PAT$GL_RLOC_BUF[DSC$W_LENGTH]) - BRB_INS_SIZ;
```

```
;  1293          3946  3 IF (.BR_DISPLACEMENT LEQ MAX_BYTE_DISP) AND (.BR_DISPLACEMENT GEQ MIN_BYTE_DISP)
;  1294          3947  2 THEN
;  1295          3948  2         BEGIN
;  1296          3949  2         INSTRUC_BUF[0] = BRB_INS_SIZ;
;  1297          3950  2         INSTRUC_BUF[1] = BRB_OPCODE;
;  1298          3951  2         CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[2],0));
;  1299          3952  2         END
;  1300          3953  2 ELSE
;  1301          3954  2         IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)
;  1302          3955  2         THEN
;  1303          3956  3                 BEGIN
;  1304          3957  3                 INSTRUC_BUF[0] = BRW_INS_SIZ;
;  1305          3958  3                 INSTRUC_BUF[1] = BRW_OPCODE;
;  1306          3959  3                 BR_DISPLACEMENT = .BR_DISPLACEMENT - (BRW_INS_SIZ - BRB_INS_SIZ);
;  1307          3960  3                 CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[2],0));
;  1308          3961  3                 END
;  1309          3962  2         ELSE
;  1310          3963  3                 BEGIN
;  1311          3964  3                 INSTRUC_BUF[0] = JMP_INS_SIZ;
;  1312          3965  3                 INSTRUC_BUF[1] = JMP_OPCODE;
;  1313          3966  3                 INSTRUC_BUF[2] = PC_DEFERRED;
;  1314          3967  3                 BR_DISPLACEMENT = .BR_DISPLACEMENT - (JMP_INS_SIZ - BRB_INS_SIZ);
;  1315          3968  3                 CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[3],0));
;  1316          3969  3                 END;
;  1317          3970  2 PAT$FILL_BUF(PAT$GL_RLOC_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]);
;  1318          3971  2
;  1319          3972  2 !++
;  1320          3973  2 ! Now insert all new instructions into the patch area.
;  1321          3974  2 !--
;  1322          3975  2 IF (.PAT$GL_RLOC_BUF[DSC$W_LENGTH] GTR .PAT$GL_PATAREA[DSC$W_LENGTH])
;  1323          3976  2 THEN
;  1324          3977  3         BEGIN
;  1325          3978  4         IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
;  1326          3979  3         THEN
;  1327          3980  4                 BEGIN
;  1328          3981  4                 PAT$EXP_AREA((.PAT$GL_RLOC_BUF[DSC$W_LENGTH] + A_PAGE - 1)/A_PAGE, .OLD_LOC);
;  1329          3982  5                 IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_RLOC_BUF[DSC$W_LENGTH])
;  1330          3983  4                 THEN
;  1331          3984  4                         SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_RLOC_BUF[DSC$W_LENGTH],
;  1332          3985  4                          .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  1333          3986  4                 END
;  1334          3987  3         ELSE
;  1335          3988  3                 SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_RLOC_BUF[DSC$W_LENGTH],
;  1336          3989  3                  .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  1337          3990  2         END;
;  1338          3991  2 PAT$WRITE_MEM(.PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_RLOC_BUF[DSC$A_POINTER], .PAT$GL_RLOC_BUF[DSC$W_LENGTH
;  1339          3992  2 PAT$GL_PATAREA[DSC$W_LENGTH] = .PAT$GL_PATAREA[DSC$W_LENGTH] - .PAT$GL_RLOC_BUF[DSC$W_LENGTH];
;  1340          3993  2 PAT$GL_PATAREA[DSC$A_POINTER] = .PAT$GL_PATAREA[DSC$A_POINTER] + .PAT$GL_RLOC_BUF[DSC$W_LENGTH];
;  1341          3994  2
;  1342          3995  2 !++
;  1343          3996  2 ! Now there is room for the branch instruction at the old location hole.
;  1344          3997  2 ! Set up a buffer with the encoded branch instruction followed by NOP's to
;  1345          3998  2 ! insert there.  Then write it to the old location hole.
;  1346          3999  2 !--
;  1347          4000  3 IF (.HOLE_SIZE GTR .BR_INSTRUC[0])
;  1348          4001  2 THEN
;  1349          4002  3         BEGIN
```

```
;  1350        4003  1          NEW_INS_PTR = PAT$FREEZ((.HOLE_SIZE + A_LONGWORD - 1)/A_LONGWORD);
;  1351        4004  1          CH$COPY(.BR_INSTRUC[0], CH$PTR(BR_INSTRUC[1], 0), NOP_INSTR,
;  1352        4005  1                  .HOLE_SIZE, CH$PTR(.NEW_INS_PTR, 0));
;  1353        4006  1          PAT$WRITE_MEM(.OLD_LOC, CH$PTR(.NEW_INS_PTR, 0), .HOLE_SIZE);
;  1354        4007  1          PAT$FREERELEASE(CH$PTR(.NEW_INS_PTR, 0), (.HOLE_SIZE + 3)/4);
;  1355        4008  1          END
;  1356        4009  1  ELSE
;  1357        4010  1          PAT$WRITE_MEM(.OLD_LOC, CH$PTR(BR_INSTRUC[1], 0), .HOLE_SIZE);
;  1358        4011  1
;  1359        4012  1  !++
;  1360        4013  1  ! Now write out all the new instructions deposited.
;  1361        4014  1  !--
;  1362        4015  1  NEXT_LOC = .OLD_LOC;
;  1363        4016  1  WHILE (.NEXT_LOC LSS .OLD_LOC+.HOLE_SIZE)
;  1364        4017  1  DO
;  1365        4018  1          BEGIN
;  1366        4019  1          PAT$OUT_MEM_LOC(.NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, NO_CASE_TABLE);
;  1367        4020  1          NEXT_LOC = .PAT$GL_NEXT_LOC;
;  1368        4021  1          END;
;  1369        4022  1  NEXT_LOC = .NEW_LOC;
;  1370        4023  1  WHILE (.NEXT_LOC LSS .PAT$GL_PATAREA[DSC$A_POINTER])
;  1371        4024  1  DO
;  1372        4025  1          BEGIN
;  1373        4026  1          PAT$OUT_MEM_LOC(.NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, NO_CASE_TABLE);
;  1374        4027  1          NEXT_LOC = .PAT$GL_NEXT_LOC;
;  1375        4028  1          END;
;  1376        4029  1
;  1377        4030  2  RETURN;
;  1378        4031  1  END;                                                    ! End of RELOCAT_INS


                                 0FFC 00000 RELOCAT_INS:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 3661
                          5B 00000000G   EF  9E 00002       MOVAB   PAT$GL_RLOC_BUF, R11
                          5E      FF50   CE  9E 00009       MOVAB   -176(SP), SP
                00000000G EF             10  BB 0000E       BISB2   #16, PAT$GL_CONTEXT+2                 ; 3762
                00000000G EF 00000000G   EF  D0 00015       MOVL    PAT$GL_RLCLABLS, PAT$GL_SYMTBPTR      ; 3763
                          50 00000000G   EF  D0 00020       MOVL    PAT$GL_PATAREA, R0                    ; 3771
                          52 00000000G   EF  3C 00027       MOVZWL  PAT$GL_TEMP_BUF, R2
                          52             60  B1 0002E       CMPW    (R0), R2
                                         5B  1E 00031       BGEQU   3$
                          51 00000000G   EF  D0 00033       MOVL    PAT$GL_IHPPTR, R1                     ; 3774
                       14 A1       04    A0  D1 0003A       CMPL    4(R0), 20(R1)
                                         36  12 0003F       BNEQ    1$
                                   04    AC  DD 00041       PUSHL   OLD_LOC                               ; 3777
                          52       01FF   C2  9E 00044       MOVAB   511(R2), R2
                       7E 52 00000200    8F  C7 00049       DIVL3   #512, R2, -(SP)
                00000000G EF             02  FB 00051       CALLS   #2, PAT$EXP_AREA
                          50 00000000G   EF  D0 00058       MOVL    PAT$GL_PATAREA, R0                    ; 3778
                00000000G EF             60  B1 0005F       CMPW    (R0), PAT$GL_TEMP_BUF
                                         26  1E 00066       BGEQU   3$
                          7E             60  3C 00068       MOVZWL  (R0), -(SP)                           ; 3781
                                   04    A0  DD 0006B       PUSHL   4(R0)
                          7E 00000000G   EF  3C 0006E       MOVZWL  PAT$GL_TEMP_BUF, -(SP)                ; 3780
```

```
                                                08  11  00075          BRB     2$                                                      ;
                                        7E      60  3C  00077  1$:     MOVZWL  (R0), -(SP)                                             ; 3785
                                            04  A0  DD  0007A          PUSHL   4(R0)                                                   ;
                                                52  DD  0007D          PUSHL   R2                                                      ; 3784
                                                03  DD  0007F  2$:     PUSHL   #3                                                      ;
                                    006D80C2    8F  DD  00081          PUSHL   #7176386                                                ;
                        00000000G   00          05  FB  00087          CALLS   #5, LIB$SIGNAL                                          ;
                                    50  00000000G EF  D0  0008E  3$:     MOVL    PAT$GL_PATAREA, R0                                      ; 3791
                                    56      04  A0  D0  00095          MOVL    4(R0), NEW_LOC                                          ;
                                    59      04  AC  D0  00099          MOVL    OLD_LOC, R9                                             ; 3797
                                50  56          59  C3  0009D          SUBL3   R9, NEW_LOC, R0                                         ;
                                  04    AE      FE  A0  9E  000A1      MOVAB   -2(R0), BR_DISPLACEMENT                                 ;
                                0000007F  8F      04  AE  D1  000A6      CMPL    BR_DISPLACEMENT, #127                                   ; 3798
                                                12  14  000AE          BGTR    4$                                                      ;
                                FFFFFF80  8F      04  AE  D1  000B0      CMPL    BR_DISPLACEMENT, #-128                                  ;
                                                08  19  000B8          BLSS    4$                                                      ;
                                    F8  AD    1102  8F  B0  000BA      MOVW    #4354, BR_INSTRUC                                       ; 3801
                                                1D  11  000C0          BRB     5$                                                      ; 3803
                                00007FFF  8F      04  AE  D1  000C2  4$:     CMPL    BR_DISPLACEMENT, #32767                         ; 3806
                                                1F  14  000CA          BGTR    6$                                                      ;
                                FFFF8000  8F      04  AE  D1  000CC      CMPL    BR_DISPLACEMENT, #-32768                                ;
                                                15  19  000D4          BLSS    6$                                                      ;
                                    F8  AD    3103  8F  B0  000D6      MOVW    #12547, BR_INSTRUC                                      ; 3809
                                        04  AE      D7  000DC          DECL    BR_DISPLACEMENT                                         ; 3811
                                    50      F8  AD  9A  000DF  5$:     MOVZBL  BR_INSTRUC, R0                                          ; 3812
                FA  AD          04  AE          50  28  000E3          MOVC3   R0, BR_DISPLACEMENT, BR_INSTRUC+2                       ;
                                                18  11  000E9          BRB     7$                                                      ; 3806
                                    F8  AD    1706  8F  B0  000EB  6$:     MOVW    #5894, BR_INSTRUC                               ; 3816
                                    FA  AD          11  8E  000F1      MNEGB   #17, BR_INSTRUC+2                                       ; 3818
                                        04  AE      04  C2  000F5      SUBL2   #4, BR_DISPLACEMENT                                     ; 3819
                                    50      F8  AD  9A  000F9          MOVZBL  BR_INSTRUC, R0                                          ; 3820
                            FB  AD          04  AE  50  28  000FD      MOVC3   R0, BR_DISPLACEMENT, BR_INSTRUC+3                       ;
                            50              59      08  AC  C1  00103  7$:     ADDL3   HOLE_SIZE, R9, R0                               ; 3828
                                            5A          50  D0  00108      MOVL    R0, NEXT_LOC                                    ;
                                            6E          50  D0  0010B      MOVL    R0, NEXT_PC                                     ; 3829
                                            55  00000000G   EF  3C  0010E      MOVZWL  PAT$GL_TEMP_BUF, SUCC_OLD_INS                   ; 3830
        08  AC          FB  AD              08              00  ED  00115  8$:     CMPZV   #0, #8, BR_INSTRUC, HOLE_SIZE           ; 3831
                                                        03  14  0011C          BGTR    9$                                              ;
                                                    00E7    31  0011E          BRW     15$                                             ;
                                                        7E  D4  00121  9$:     CLRL    -(SP)                                           ; 3839
                                            00000000G   EF  9F  00123          PUSHAB  PAT$GL_OLD_ASD                                  ;
                                            00000000'   EF  9F  00129          PUSHAB  OLD_TAB_STG                                     ;
                                                        5A  DD  0012F          PUSHL   NEXT_LOC                                        ;
                                    00000000V   EF          04  FB  00131      CALLS   #4, PAT$OUT_MEM_LOC                             ;
                                            00000000G   EF  D4  00138          CLRL    PAT$GL_BUF_SIZ                                  ; 3840
                                    00000000G   EF      59  AE  9E  0013E      MOVAB   LOCAL_BUF+1, PAT$CP_OUT_STR                     ; 3841
                                                        7E  D4  00146          CLRL    -(SP)                                           ; 3852
                                            00000000G   EF  9F  00148          PUSHAB  PAT$GL_OLD_ASD                                  ;
                                                    08  AE  9F  0014E          PUSHAB  NEXT_PC                                         ;
                                                        7E  D4  00151          CLRL    -(SP)                                           ;
                                                        5A  DD  00153          PUSHL   NEXT_LOC                                        ;
                                    00000000G   EF          05  FB  00155      CALLS   #5, PAT$INS_DECODE                              ;
                                                6E          50  D0  0015C      MOVL    R0, NEXT_PC                                     ;
                                                            0D  12  0015F      BNEQ    10$                                             ;
                                            006D8102        8F  DD  00161      PUSHL   #7176450                                        ; 3854
                                    00000000G   00          01  FB  00167      CALLS   #1, LIB$SIGNAL                                  ;
                                            58  AE  00000000G   EF  90  0016E  10$:     MOVB    PAT$GL_BUF_SIZ, LOCAL_BUF           ; 3855
```

```
                                57         08    AE   9E 00176          MOVAB   INSTRUC_BUF, NEW_INS_PTR                    ; 3856
                            00000000G      EF    9F 0017A          PUSHAB   PAT$GL_TEMP_BUF                               ; 3857
                            00000000G      EF    9F 00180          PUSHAB   PAT$GL_NEW_ASD
                                50 00000000G   EF    3C 00186          MOVZWL   PAT$GL_TEMP_BUF, R0                        ; 3858
                                         6049    9F 0018D          PUSHAB   (R0)[R9]
                                        14       AE   9F 00190          PUSHAB   INSTRUC_BUF                               ; 3857
                                        68       AE   9F 00193          PUSHAB   LOCAL_BUF
                        00000000G       EF             05   FB 00196          CALLS   #5, PAT$INS_ENCODE
                                23                     50   E8 00199          BLBS    R0, 12$
                                    00000000G      EF    95 001A0          TSTB    PAT$GB_SUBST_IN                          ; 3860
                                                   09    13 001A6          BEQL    11$
                                57 00000000G       EF    9E 001A8          MOVAB   PAT$GB_SUBST_IN, NEW_INS_PTR             ; 3862
                                                   12    11 001AF          BRB     12$
                                        58       AE   9F 001B1 11$:      PUSHAB   LOCAL_BUF                                ; 3864
                                                   01    DD 001B4          PUSHL   #1
                                    006D810A       8F    DD 001B6          PUSHL   #7176458
                        00000000G       00             03   FB 001BC          CALLS   #3, LIB$SIGNAL
                                8F      8F         01    A7   91 001C3 12$:      CMPB    1(NEW_INS_PTR), #143             ; 3868
                                                   0E    13 001C8          BEQL    13$
                                AF      8F         01    A7   91 001CA          CMPB    1(NEW_INS_PTR), #175             ; 3869
                                                   07    13 001CF          BEQL    13$
                                CF      8F         01    A7   91 001D1          CMPB    1(NEW_INS_PTR), #207             ; 3870
                                                   0D    12 001D6          BNEQ    14$
                                    006D82CA       8F    DD 001D8 13$:      PUSHL   #7176906                           ; 3872
                        00000000G       00             01   FB 001DE          CALLS   #1, LIB$SIGNAL
                                        7E             67   9A 001E5 14$:      MOVZBL  (NEW_INS_PTR), -(SP)               ; 3873
                                            01         A7   9F 001E8          PUSHAB  1(NEW_INS_PTR)
                                    00000000G      EF    9F 001EB          PUSHAB  PAT$GL_TEMP_BUF
                        00000000V       EF             03   FB 001F1          CALLS   #3, PAT$FILL_BUF
                                50          08     AC   6E   C1 001F8          ADDL3   NEXT_PC, HOLE_SIZE, R0             ; 3874
                            08  AC          50            5A   C3 001FD          SUBL3   NEXT_LOC, R0, HOLE_SIZE
                                            5A            6E   D0 00202          MOVL    NEXT_PC, NEXT_LOC                  ; 3875
                                                    FF0D   31   00205          BRW     8$                                 ; 3831
                                53 00000000G       EF    D0 00208 15$:      MOVL    PAT$GL_TEMP_BUF+4, CUR_LOC         ; 3882
                                6E                     59   D0 0020F          MOVL    R9, NEXT_PC                        ; 3883
                                52             10      AC   D0 00212          MOVL    ASC_INS_PTR, NXT_ASC_INS           ; 3884
                                50 00000000G       EF    D0 00216 16$:      MOVL    PAT$GL_TEMP_BUF+4, R0              ; 3885
                                51 00000000G       EF    3C 0021D          MOVZWL  PAT$GL_TEMP_BUF, R1
                                51                     50   C0 00224          ADDL2   R0, R1
                                51                     53   D1 00227          CMPL    CUR_LOC, R1
                                                       03   1F 0022A          BLSSU   17$
                                                    00EB   31   0022C          BRW     25$
                                    00000000G      EF    D4 0022F 17$:      CLRL    PAT$GL_BUF_SIZ                     ; 3888
                        00000000G       EF             59   AE   9E 00235          MOVAB   LOCAL_BUF+1, PAT$CP_OUT_STR        ; 3889
                    51                  50         0C    AC   C1 0023D          ADDL3   OLD_INS_SIZ, R0, R1                ; 3890
                                        51                53   D1 00242          CMPL    CUR_LOC, R1
                                                          1C   1F 00245          BLSSU   18$
                                        50                55   C0 00247          ADDL2   SUCC_OLD_INS, R0                   ; 3891
                                        50                53   D1 0024A          CMPL    CUR_LOC, R0
                                                          14   1E 0024D          BGEQU   18$
                                        54             04    A2   D0 0024F          MOVL    4(NXT_ASC_INS), DECODED_INS        ; 3900
                                        52                62   D0 00253          MOVL    (NXT_ASC_INS), NXT_ASC_INS         ; 3901
                            00000000G   EF 00000000G       EF   D0 00256          MOVL    PAT$GL_RCLABLS, PAT$GL_SYMTBPTR    ; 3902
                                                          0F   11 00261          BRB     19$                                ; 3890
                                        54             58    AE   9E 00263 18$:      MOVAB   LOCAL_BUF, DECODED_INS     ; 3910
                            00000000G   EF 00000000G       EF   D0 00267          MOVL    PAT$GL_OLDLABLS, PAT$GL_SYMTBPTR   ; 3911
                                                          7E   D4 00272 19$:      CLRL    -(SP)                              ; 3913
```

```
                        00000000G   EF   9F  00274          PUSHAB  PAT$GL_NEW_ASD                                          ;
                                08  AE   9F  0027A          PUSHAB  NEXT_PC                                                 ;
                                    7E   D4  0027D          CLRL    -(SP)                                                   ;
                                    53   DD  0027F          PUSHL   CUR_LOC                                                 ;
00000000G                       EF       05  FB  00281      CALLS   #5, PAT$INS_DECODE                                      ;
                                53       50  D0  00288      MOVL    R0, CUR_LOC                                             ;
                                         0D  12  0028B      BNEQ    20$                                                     ;
                            006D8102     8F   DD  0028D     PUSHL   #7176450                                                ; 3915
00000000G                   00           01   FB  00293     CALLS   #1, LIB$SIGNAL                                          ;
                    58      AE 00000000G EF   90  0029A 20$: MOVB    PAT$GL_BUF_SIZ, LOCAL_BUF                               ; 3916
                    57              08   AE   9E  002A2     MOVAB   INSTRUC_BUF, NEW_INS_PTR                                ; 3917
                                    5B   DD  002A6          PUSHL   R11                                                     ; 3918
                            00000000G    EF   9F  002A8     PUSHAB  PAT$GL_NEW_ASD                                          ;
                                50       6B   3C  002AE     MOVZWL  PAT$GL_RLOC_BUF, R0                                     ; 3919
                                      6046   9F  002B1      PUSHAB  (R0)[NEW_LOC]                                           ;
                                    14   AE   9F  002B4     PUSHAB  INSTRUC_BUF                                             ; 3918
                                    54   DD  002B7          PUSHL   DECODED_INS                                             ;
00000000G                       EF       05  FB  002B9      CALLS   #5, PAT$INS_ENCODE                                      ;
                                23       50  E8  002C0      BLBS    R0, 22$                                                 ;
                            00000000G    EF   95  002C3     TSTB    PAT$GB_SUBST_IN                                         ; 3921
                                    09   13  002C9          BEQL    21$                                                     ;
                    57      00000000G    EF   9E  002CB     MOVAB   PAT$GB_SUBST_IN, NEW_INS_PTR                            ; 3923
                                    12   11  002D2          BRB     22$                                                     ;
                                    58   AE   9F  002D4 21$: PUSHAB  LOCAL_BUF                                             ; 3925
                                    01   DD  002D7          PUSHL   #1                                                      ;
                            006D810A     8F   DD  002D9     PUSHL   #7176458                                                ;
00000000G                   00           03   FB  002DF     CALLS   #3, LIB$SIGNAL                                          ;
                    8F      8F       01  A7   91  002E6 22$: CMPB    1(NEW_INS_PTR), #143                                    ; 3929
                                    0E   13  002EB          BEQL    23$                                                     ;
                    AF      8F       01  A7   91  002ED     CMPB    1(NEW_INS_PTR), #175                                    ; 3930
                                    07   13  002F2          BEQL    23$                                                     ;
                    CF      8F       01  A7   91  002F4     CMPB    1(NEW_INS_PTR), #207                                    ; 3931
                                    0D   12  002F9          BNEQ    24$                                                     ;
                            006D82CA     8F   DD  002FB 23$: PUSHL   #7176906                                                ; 3933
                        00000000G   00       01   FB  00301 CALLS   #1, LIB$SIGNAL                                          ;
                                7E           67   9A  00308 24$: MOVZBL  (NEW_INS_PTR), -(SP)                               ; 3934
                                        01   A7   9F  0030B     PUSHAB  1(NEW_INS_PTR)                                          ;
                                    5B   DD  0030E          PUSHL   R11                                                     ;
                        00000000V   EF       03   FB  00310 CALLS   #3, PAT$FILL_BUF                                      ;
                                      FEFC   31  00317      BRW     16$                                                     ; 3885
                        00000000G   EF 00000000G EF   D0  0031A 25$: MOVL    PAT$GL_RLCLABLS, PAT$GL_SYMTBPTR                ; 3936
                                    5B   DD  00325          PUSHL   R11                                                     ; 3937
                        00000000G   EF       01   FB  00327 CALLS   #1, PAT$RESOLVE_INS                                   ;
                                50 00000000G EF   D0  0032E     MOVL    PAT$GL_PATAREA, R0                                  ; 3944
                                51       6B   3C  00335     MOVZWL  PAT$GL_RLOC_BUF, R1                                     ; 3945
                    50          51       04   A0  C1  00338 ADDL3   4(R0), R1, R0                                          ;
                                50           5A  C2  0033D SUBL2   NEXT_LOC, R0                                            ; 3944
                            04  AE           50  CE  00340 MNEGL   R0, BR_DISPLACEMENT                                     ; 3945
                            04  AE           02  C2  00344 SUBL2   #2, BR_DISPLACEMENT                                     ;
                    0000007F    8F       04  AE   D1  00348 CMPL    BR_DISPLACEMENT, #127                                   ; 3946
                                    12   14  00350          BGTR    26$                                                     ;
                    FFFFFF80    8F       04  AE   D1  00352 CMPL    BR_DISPLACEMENT, #-128                                  ;
                                    08   19  0035A          BLSS    26$                                                     ;
                            08  AE       1102  8F   B0  0035C     MOVW    #4354, INSTRUC_BUF                                  ; 3949
                                    1D   11  00362          BRB     27$                                                     ; 3951
                    00007FFF    8F       04  AE   D1  00364 26$: CMPL    BR_DISPLACEMENT, #32767                         ; 3954
                                    1F   14  0036C          BGTR    28$                                                     ;
```

```
                FFFF8000    8F      04    AE  D1 0036E          CMPL    BR_DISPLACEMENT, #-32768                        ;
                                          15  19 00376          BLSS    28$                                             ;
                      08    AE    3103    8F  B0 00378          MOVW    #12547, INSTRUC_BUF                             ; 3957
                                    04    AE  D7 0037E          DECL    BR_DISPLACEMENT                                 ; 3959
                            50      08    AE  9A 00381  27$:    MOVZBL  INSTRUC_BUF, R0                                 ; 3960
          0A    AE      04    AE            50  28 00385          MOVC3   R0, BR_DISPLACEMENT, INSTRUC_BUF+2              ;
                                          18  11 0038B          BRB     29$                                             ; 3954
                      08    AE    1706    8F  B0 0038D  28$:    MOVW    #5894, INSTRUC_BUF                              ; 3964
                      0A    AE            11  8E 00393          MNEGB   #17, INSTRUC_BUF+2                              ; 3966
                      04    AE            04  C2 00397          SUBL2   #4, BR_DISPLACEMENT                             ; 3967
                            50      08    AE  9A 0039B          MOVZBL  INSTRUC_BUF, R0                                 ; 3968
          0B    AE      04    AE            50  28 0039F          MOVC3   R0, BR_DISPLACEMENT, INSTRUC_BUF+3              ;
                            7E      08    AE  9A 003A5  29$:    MOVZBL  INSTRUC_BUF, -(SP)                              ; 3970
                                    0D    AE  9F 003A9          PUSHAB  INSTRUC_BUF+1                                   ;
                                          5B  DD 003AC          PUSHL   R11                                             ;
                00000000V   EF              03  FB 003AE          CALLS   #3, PAT$FILL_BUF                                ;
                            52              6B  3C 003B5          MOVZWL  PAT$GL_RLOC_BUF, R2                             ; 3975
                            50  00000000G    EF  D0 003B8          MOVL    PAT$GL_PATAREA, R0                              ;
                            52              60  B1 003BF          CMPW    (R0), R2                                        ;
                                            52  1E 003C2          BGEQU   32$                                             ;
                            51  00000000G    EF  D0 003C4          MOVL    PAT$GL_IHPPTR, R1                               ; 3978
                      14    A1        04    A0  D1 003CB          CMPL    4(R0), 20(R1)                                   ;
                                            2D  12 003D0          BNEQ    30$                                             ;
                                            59  DD 003D2          PUSHL   R9                                              ; 3981
                            52      01FF    C2  9E 003D5          MOVAB   511(R2), R2                                     ;
                7E          52  00000200    8F  C7 003D9          DIVL3   #512, R2, -(SP)                                 ;
                00000000G   EF              02  FB 003E1          CALLS   #2, PAT$EXP_AREA                                ;
                            50  00000000G    EF  D0 003E8          MOVL    PAT$GL_PATAREA, R0                              ; 3982
                            6B              60  B1 003EF          CMPW    (R0), PAT$GL_RLOC_BUF                           ;
                                            22  1E 003F2          BGEQU   32$                                             ;
                            7E              60  3C 003F4          MOVZWL  (R0), -(SP)                                     ; 3985
                                    04      A0  DD 003F7          PUSHL   4(R0)                                           ;
                            7E              6B  3C 003FA          MOVZWL  PAT$GL_RLOC_BUF, -(SP)                          ; 3984
                                            08  11 003FD          BRB     31$                                             ;
                            7E              60  3C 003FF  30$:    MOVZWL  (R0), -(SP)                                     ; 3989
                                    04      A0  DD 00402          PUSHL   4(R0)                                           ;
                                            52  DD 00405          PUSHL   R2                                              ; 3988
                                            03  DD 00407  31$:    PUSHL   #3                                              ;
                            006D80C2        8F  DD 00409          PUSHL   #7176386                                        ;
                00000000G   00              05  FB 0040F          CALLS   #5, LIB$SIGNAL                                  ;
                            7E              6B  3C 00416  32$:    MOVZWL  PAT$GL_RLOC_BUF, -(SP)                          ; 3991
                                    04      AB  DD 00419          PUSHL   PAT$GL_RLOC_BUF+4                               ;
                            50  00000000G    EF  D0 0041C          MOVL    PAT$GL_PATAREA, R0                              ;
                                    04      A0  DD 00423          PUSHL   4(R0)                                           ;
                00000000G   EF              03  FB 00426          CALLS   #3, PAT$WRITE_MEM                               ;
                            50  00000000G    EF  D0 0042D          MOVL    PAT$GL_PATAREA, R0                              ; 3992
                            60              6B  A2 00434          SUBW2   PAT$GL_RLOC_BUF, (R0)                           ;
                            51              6B  3C 00437          MOVZWL  PAT$GL_RLOC_BUF, R1                             ; 3993
                      04    A0              51  C0 0043A          ADDL2   R1, 4(R0)                                       ;
                            58      08      AC  D0 0043E          MOVL    HOLE_SIZE, R8                                   ; 4000
    58      F8    AD            08          00  ED 00442          CMPZV   #0, #8, BR_INSTRUC, R8                          ;
                                            3C  18 00448          BGEQ    33$                                             ;
                            50      03      A8  9E 0044A          MOVAB   3(R8), R0                                       ; 4003
                7E          50              04  C7 0044E          DIVL3   #4, R0, -(SP)                                   ;
                00000000G   EF              01  FB 00452          CALLS   #1, PAT$FREEZ                                   ;
                            57              50  D0 00459          MOVL    R0, NEW_INS_PTR                                 ;
                            50      F8      AD  9A 0045C          MOVZBL  BR_INSTRUC, R0                                  ; 4004
```

```
                58              01      F9      AD              50  2C 00460            MOVC5   R0, BR_INSTRUC+1, #1, R8, (NEW_INS_PTR)        ; 4005
                                                                67     00466
                                        7E                      57  7D 00467            MOVQ    NEW_INS_PTR, -(SP)                              ; 4006
                                                                59  DD 0046A            PUSHL   R9
                                    00000000G   EF              03  FB 0046C            CALLS   #3, PAT$WRITE_MEM
                                                50          03  A8  9E 00473            MOVAB   3(R8), R0                                       ; 4007
                                7E              50              04  C7 00477            DIVL3   #4, R0, -(SP)
                                                                57  DD 0047B            PUSHL   NEW_INS_PTR
                                    00000000G   EF              02  FB 0047D            CALLS   #2, PAT$FREERELEASE
                                                                0E  11 00484            BRB     34$                                             ; 4000
                                                                58  DD 00486 33$:       PUSHL   R8                                              ; 4010
                                                        F9      AD  9F 00488            PUSHAB  BR_INSTRUC+1
                                                                59  DD 0048B            PUSHL   R9
                                    00000000G   EF              03  FB 0048D            CALLS   #3, PAT$WRITE_MEM
                                                5A              59  D0 00494 34$:       MOVL    R9, NEXT_LOC                                    ; 4015
                                                59              58  C0 00497            ADDL2   R8, R9                                          ; 4016
                                                59              5A  D1 0049A 35$:       CMPL    NEXT_LOC, R9
                                                                20  18 0049D            BGEQ    36$
                                                                7E  D4 0049F            CLRL    -(SP)                                           ; 4019
                                                    00000000G   EF  9F 004A1            PUSHAB  PAT$GL_NEW_ASD
                                                    00000000'   EF  9F 004A7            PUSHAB  NEW_TAB_STG
                                                                5A  DD 004AD            PUSHL   NEXT_LOC
                                    00000000V   EF              04  FB 004AF            CALLS   #4, PAT$OUT_MEM_LOC
                                                5A 00000000G    EF  D0 004B6            MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                       ; 4020
                                                                DB  11 004BD            BRB     35$                                             ; 4016
                                                5A              56  D0 004BF 36$:       MOVL    NEW_LOC, NEXT_LOC                               ; 4022
                                                50 00000000G    EF  D0 004C2 37$:       MOVL    PAT$GL_PATAREA, R0                              ; 4023
                                        04      A0                  5A  D1 004C9            CMPL    NEXT_LOC, 4(R0)
                                                                20  18 004CD            BGEQ    38$
                                                                7E  D4 004CF            CLRL    -(SP)                                           ; 4026
                                                    00000000G   EF  9F 004D1            PUSHAB  PAT$GL_NEW_ASD
                                                    00000000'   EF  9F 004D7            PUSHAB  NEW_TAB_STG
                                                                5A  DD 004DD            PUSHL   NEXT_LOC
                                    00000000V   EF              04  FB 004DF            CALLS   #4, PAT$OUT_MEM_LOC
                                                5A 00000000G    EF  D0 004E6            MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                       ; 4027
                                                                D3  11 004ED            BRB     37$                                             ; 4023
                                                                    04 004EF 38$:       RET                                                     ; 4031

; Routine Size:  1264 bytes,    Routine Base:  _PAT$CODE + 066B
```

```
;  1380     4032  1 GLOBAL ROUTINE PAT$SUBST_INS (OLD_INS_PTR, INS_PC) =
;  1381     4033  1
;  1382     4034  1 !++
;  1383     4035  1 ! FUNCTIONAL DESCRIPTION:
;  1384     4036  1 !
;  1385     4037  1 ! This routine substitutes other instruction sequences for branch-type
;  1386     4038  1 ! instructions that have been relocated to a new address and whose branch
;  1387     4039  1 ! displacements are now too small.  The following table describes the
;  1388     4040  1 ! possible substitutions.  If the branch in the first replacement choice does
;  1389     4041  1 ! not reach, then the second replacement choice is used.  Notice that the blank
;  1390     4042  1 ! lines in the table separate groups of instructions that are handled
;  1391     4043  1 ! similarly for substitutions.
;  1392     4044  1 !
;  1393     4045  1 ! OPC   INSTRUC          REPLACEMENT 1                  REPLACEMENT 2
;  1394     4046  1 ! ---   -------          -------------                  -------------
;  1395     4047  1 !
;  1396     4048  1 ! 12    BNEQ <X>         BEQL .+03, BRW <X>             BEQL .+06, JMP <X>
;  1397     4049  1 ! 13    BEQL <X>         BNEQ .+03, BRW <X>             BNEQ .+06, JMP <X>
;  1398     4050  1 ! 14    BGTR <X>         BLEQ .+03, BRW <X>             BLEQ .+06, JMP <X>
;  1399     4051  1 ! 15    BLEQ <X>         BGTR .+03, BRW <X>             BGTR .+06, JMP <X>
;  1400     4052  1 ! 18    BGEQ <X>         BLSS .+03, BRW <X>             BLSS .+06, JMP <X>
;  1401     4053  1 ! 19    BLSS <X>         BLSS .+03, BRW <X>             BLSS .+06, JMP <X>
;  1402     4054  1 ! 1A    BGTRU <X>        BLEQU .+03, BRW <X>            BLEQU .+06, JMP <X>
;  1403     4055  1 ! 1B    BLEQU <X>        BGTRU .+03, BRW <X>            BGTRU .+06, JMP <X>
;  1404     4056  1 ! 1C    BVC <X>          BVS .+03, BRW <X>              BVS .+06, JMP <X>
;  1405     4057  1 ! 1D    BVS <X>          BVC .+03, BRW <X>              BVC .+06, JMP <X>
;  1406     4058  1 ! 1E    BGEQU <X>        BLSSU .+03, BRW <X>            BLSSU .+06, JMP <X>
;  1407     4059  1 ! 1F    BLSSU <X>        BGEQU .+03, BRW <X>            BGEQU .+06, JMP <X>
;  1408     4060  1 ! E0    BBS <X>          BBC .+03, BRW <X>              BBC .+06, JMP <X>
;  1409     4061  1 ! E1    BBC <X>          BBS .+03, BRW <X>              BBS .+06, JMP <X>
;  1410     4062  1 ! E2    BBSS <X>         BBCS .+03, BRW <X>             BBCS .+06, JMP <X>
;  1411     4063  1 ! E3    BBCS <X>         BBSS .+06, BRW <X>             BBSS .+03, JMP <X>
;  1412     4064  1 ! E4    BBSC <X>         BBCC .+03, BRW <X>             BBCC .+06, JMP <X>
;  1413     4065  1 ! E5    BBCC <X>         BBSC .+03, BRW <X>             BBSC .+06, JMP <X>
;  1414     4066  1 ! E8    BLBS <X>         BLBC .+03, BRW <X>             BLBC .+06, JMP <X>
;  1415     4067  1 ! E9    BLBC <X>         BLBS .+03, BRW <X>             BLBS .+06, JMP <X>
;  1416     4068  1 !
;  1417     4069  1 ! E6    BBSSI <X>        BBSSI .+02, BRB .+03, BRW <X>  BBSSI .+02, BRB .+06, JMP <X>
;  1418     4070  1 ! E7    BBCCI <X>        BBCCI .+02, BRB .+03, BRW <X>  BBCCI .+02, BRB .+06, JMP <X>
;  1419     4071  1 ! F2    AOBLSS <X>       AOBLSS .+02, BRB .+03, BRW <X> AOBLSS .+02, BRB .+06, JMP <X>
;  1420     4072  1 ! F3    AOBLEQ <X>       AOBLEQ .+02, BRB .+03, BRW <X> AOBLEQ .+02, BRB .+06, JMP <X>
;  1421     4073  1 ! F4    SOBGEQ <X>       SOBGEQ .+02, BRB .+03, BRW <X> SOBGEQ .+02, BRB .+06, JMP <X>
;  1422     4074  1 ! F5    SOBGTR <X>       SOBGTR .+02, BRB .+03, BRW <X> SOBGTR .+02, BRB .+06, JMP <X>
;  1423     4075  1 !
;  1424     4076  1 ! 9D    ACBB <X>         ACBB .+02, BRB .+06, JMP <X>
;  1425     4077  1 ! 3D    ACBW <X>         ACBW .+02, BRB .+06, JMP <X>
;  1426     4078  1 ! F1    ACBL <X>         ACBL .+02, BRB .+06, JMP <X>
;  1427     4079  1 ! 4F    ACBF <X>         ACBF .+02, BRB .+06, JMP <X>
;  1428     4080  1 ! 6F    ACBD <X>         ACBD .+02, BRB .+06, JMP <X>
;  1429     4081  1 ! 4FFD  ACBG <X>         ACBG .+02, BRB .+06, JMP <X>
;  1430     4082  1 ! 6FFD  ACBH <X>         ACBH .+02, BRB .+06, JMP <X>
;  1431     4083  1 !
;  1432     4084  1 ! 11    BRB <X>          BRW <X>                        JMP <X>
;  1433     4085  1 ! 10    BSBB <X>         BSBW <X>                       JSB <X>
;  1434     4086  1 !
;  1435     4087  1 ! 31    BRW <X>          JMP <X>
;  1436     4088  1 ! 30    BSBW <X>         JSB <X>
```

```
;  1437     4089  1 !
;  1438     4090  1 ! In addition to the above instructions, there are also three case
;  1439     4091  1 ! instructions.  None of these has a replacement.  (In fact, the encoder
;  1440     4092  1 ! does not know how to insert a case instruction correctly; it will only
;  1441     4093  1 ! insert the instruction parameters.  The branch displacements must be
;  1442     4094  1 ! inserted as .WORD directives.)
;  1443     4095  1 !
;  1444     4096  1 ! CALLING SEQUENCE:
;  1445     4097  1 !
;  1446     4098  1 !       PAT$SUBST_INS (OLD-ENCODED-INSTRUCTION-ADDRESS, PC-OF-INSTRUCTION)
;  1447     4099  1 !
;  1448     4100  1 ! INPUTS:
;  1449     4101  1 !
;  1450     4102  1 !       OLD_INS_PTR - Address of counted instruction stream to be substituted
;  1451     4103  1 !       INS_PC = Unmapped address of where to put instruction
;  1452     4104  1 !
;  1453     4105  1 ! IMPLICIT INPUTS:
;  1454     4106  1 !
;  1455     4107  1 !       PAT$GB_SUBST_IN - Buffer for substitution counted byte stream
;  1456     4108  1 !
;  1457     4109  1 ! OUTPUTS:
;  1458     4110  1 !
;  1459     4111  1 !       NONE
;  1460     4112  1 !
;  1461     4113  1 ! IMPLICIT OUTPUTS:
;  1462     4114  1 !
;  1463     4115  1 !       The substitution binary stream is written into INSTRUC_BUF
;  1464     4116  1 !       as a counted byte stream.
;  1465     4117  1 !
;  1466     4118  1 ! ROUTINE VALUE:
;  1467     4119  1 !
;  1468     4120  1 !       FALSE if no substitution instructions were possible.
;  1469     4121  1 !       TRUE if substitution was successful.
;  1470     4122  1 !
;  1471     4123  1 ! SIDE EFFECTS:
;  1472     4124  1 !
;  1473     4125  1 !       A substitution stream can now be written to memory, or an error
;  1474     4126  1 !       reported.  However, if an instruction had a label associated with it
;  1475     4127  1 !       any branches elsewhere in the code to it will no longer work!!!
;  1476     4128  1 !
;  1477     4129  1 !--
;  1478     4130  1
;  1479     4131  2 BEGIN
;  1480     4132  2
;  1481     4133  2 MAP
;  1482     4134  2         OLD_INS_PTR : REF VECTOR[,BYTE];                ! Old binary instruction stream
;  1483     4135  2
;  1484     4136  2 LITERAL
;  1485     4137  2         MIN_WORD_DISP = -32768,                         ! Minimum displacement for BRW
;  1486     4138  2         MAX_WORD_DISP = 32767,                          ! Maximum displacement for BRW
;  1487     4139  2         BRB_OPCODE = %X'11',                            ! Opcode for BRB instruction
;  1488     4140  2         BRW_OPCODE = %X'31',                            ! Opcode for BRW instruction
;  1489     4141  2         JMP_OPCODE = %X'17',                            ! Opcode forJMP instruction
;  1490     4142  2         BNEQ_OPCODE = %X'12',                           ! Opcode for BNEQ instruction
;  1491     4143  2         BLEQ_OPCODE = %X'15',                           ! Opcode for BLEQ instruction
;  1492     4144  2         BGEQ_OPCODE = %X'18',                           ! Opcode for BGEQ instruction
;  1493     4145  2         BLSSU_OPCODE = %X'1F',                          ! Opcode for BLSSU instruction
```

```
;  1494          4146  2         BBS_OPCODE = %X'E0',                    ! Opcode for BBS instruction
;  1495          4147  2         BBCC_OPCODE = %X'E5',                   ! Opcode for BBCC instruction
;  1496          4148  2         BLBS_OPCODE = %X'E8',                   ! Opcode for BLBS instruction
;  1497          4149  2         BLBC_OPCODE = %X'E9',                   ! Opcode for BLBC instruction
;  1498          4150  2         BBSSI_OPCODE = %X'E6',                  ! Opcode for BBSSI instruction
;  1499          4151  2         BBCCI_OPCODE = %X'E7',                  ! Opcode for BBCCI instruction
;  1500          4152  2         AOBLSS_OPCODE = %X'F2',                 ! Opcode for AOBLSS instruction
;  1501          4153  2         SOBGTR_OPCODE = %X'F5',                 ! Opcode for SOBGTR instruction
;  1502          4154  2         ACBB_OPCODE = %X'9D',                   ! Opcode for ACBB instruction
;  1503          4155  2         ACBW_OPCODE = %X'3D',                   ! Opcode for ACBW instruction
;  1504          4156  2         ACBL_OPCODE = %X'F1',                   ! Opcode for ACBL instruction
;  1505          4157  2         ACBF_OPCODE = %X'4F',                   ! Opcode for ACBF instruction
;  1506          4158  2         ACBD_OPCODE = %X'6F',                   ! Opcode for ACBD instruction
;  1507          4159  2         ACBG_HICODE = %X'4F',                   ! High byte of Opcode for ACBG instruction
;  1508          4160  2         ACBH_HICODE = %X'6F',                   ! High byte of Opcode for ACBH instruction
;  1509          4161  2         CASEB_OPCODE = %X'8F',                  ! Opcode for CASEB instruction
;  1510          4162  2         CASEW_OPCODE = %X'AF',                  ! Opcode for CASEW instruction
;  1511          4163  2         CASEL_OPCODE = %X'CF',                  ! Opcode for CASEL instruction
;  1512          4164  2         BSBW_OPCODE = %X'30',                   ! Opcode for BSBW instruction
;  1513          4165  2         BSBB_OPCODE = %X'10',                   ! Opcode for BSBB instruction
;  1514          4166  2         JSB_OPCODE = %X'16',                    ! Opcode for JSB instruction
;  1515          4167  2         BRB_INS_SIZ = 2,                        ! Size of BRB instruction
;  1516          4168  2         BRW_INS_SIZ = 3,                        ! Size of BRW instruction
;  1517          4169  2         JMP_INS_SIZ = 6,                        ! Size of JMP instruction
;  1518          4170  2         PC_DEFERRED = %X'EF',                   ! PC deferred instruction mode
;  1519          4171  2         MAX_INST_LEN    = 80;                   ! Maximum number of binary bytes in an instr
;  1520          4172  2
;  1521          4173  2 LOCAL
;  1522          4174  2         BR_DISPLACEMENT : SIGNED LONG;          ! Displacement for branch instruction
;  1523          4175  2
;  1524          4176  2 !++
;  1525          4177  2 ! Handle the first group of substitutions.  These may be replaced with
;  1526          4178  2 ! their complement and a BRW, i.e., opcodes BGTR through BLBC in the above
;  1527          4179  2 ! table.  The complement instruction must be set to branch around the BRW
;  1528          4180  2 ! instruction.  Therefore, the instruction stream changes from:
;  1529          4181  2 !               <BR INS> TO <X>
;  1530          4182  2 ! TO:
;  1531          4183  2 !               <BR COM INS> TO .+03            BRW <X>
;  1532          4184  2 !--
;  1533          4185  2 IF (.OLD_INS_PTR[1] GEQU BNEQ_OPCODE AND .OLD_INS_PTR[1] LEQU BLEQ_OPCODE) OR
;  1534          4186  2    (.OLD_INS_PTR[1] GEQU BGEQ_OPCODE AND .OLD_INS_PTR[1] LEQU BLSSU_OPCODE) OR
;  1535          4187  2    (.OLD_INS_PTR[1] GEQU BBS_OPCODE  AND .OLD_INS_PTR[1] LEQU BBCC_OPCODE ) OR
;  1536          4188  2    (.OLD_INS_PTR[1] GEQU BLBS_OPCODE AND .OLD_INS_PTR[1] LEQU BLBC_OPCODE )
;  1537          4189  2 THEN
;  1538          4190  3         BEGIN
;  1539          4191  3         !++
;  1540          4192  3         ! Build the binary instruction stream for the complement branch.
;  1541          4193  3         ! Then build the BRW instruction with the old branch's displacement.
;  1542          4194  3         !--
;  1543          4195  3         PAT$GB_SUBST_IN[0] = BRW_INS_SIZ + .OLD_INS_PTR[0];     ! Set the entire stream length
;  1544          4196  3         PAT$GB_SUBST_IN[1] = (IF .OLD_INS_PTR[1] THEN (.OLD_INS_PTR[1] - 1)
;  1545          4197                               ELSE (.OLD_INS_PTR[1] + 1));   ! Set complement opcode
;  1546          4198  3         CH$MOVE(.OLD_INS_PTR[0]-2, CH$PTR(OLD_INS_PTR[2]), CH$PTR(PAT$GB_SUBST_IN[2])); ! Move in instructio
;  1547          4199  3         PAT$GB_SUBST_IN[.OLD_INS_PTR[0]] = BRW_INS_SIZ;         ! Set complement branch around BRW instructi
;  1548          4200  3         PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+1] = BRW_OPCODE;        ! Set BRW instruction opcode
;  1549          4201  3         BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
;  1550          4202  3         IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
```

```
;  1551          4203  4            (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)                    ! Does displacement fit
;  1552          4204  3        THEN
;  1553          4205  3            (PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ])<0,16,1> = .BR_DISPLACEMENT       ! Yes, move it into
;  1554          4206  3        ELSE
;  1555          4207  4            BEGIN
;  1556          4208  4            !++
;  1557          4209  4            ! No, it did not fit.  Use a JMP instead of a BRW, which
;  1558          4210  4            ! is the second choice in the table.  The complement branch
;  1559          4211  4            ! displacement must be changed, too.
;  1560          4212  4            !--
;  1561          4213  4            PAT$GB_SUBST_IN[0] = .PAT$GB_SUBST_IN[0] + (JMP_INS_SIZ - BRW_INS_SIZ); ! Set new instructio
;  1562          4214  4            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0] ] = JMP_INS_SIZ; ! Set complement branch around JMP instruc
;  1563          4215  4            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+1 ] = JMP_OPCODE; ! Set JMP opcode
;  1564          4216  4            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = PC_DEFERRED; ! Set instruction mode
;  1565          4217  4            (PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ])<0,32,1> = .BR_DISPLACEMENT -
;  1566          4218  4                                    (JMP_INS_SIZ - BRW_INS_SIZ); ! Set new branch displacement
;  1567          4219  4
;  1568          4220  4            END
;  1569          4221  3        END
;  1570          4222  2    ELSE
;  1571          4223  2    !++
;  1572          4224  2    ! The opcode was not one of the first group, therefore check to see if
;  1573          4225  2    ! it was one of the second group.
;  1574          4226  2    !--
;  1575          4227  2    IF (.OLD_INS_PTR[1] EQLU BBSSI_OPCODE) OR
;  1576          4228  2       (.OLD_INS_PTR[1] EQLU BBCCI_OPCODE) OR
;  1577          4229  3       (.OLD_INS_PTR[1] GEQU AOBLSS_OPCODE AND .OLD_INS_PTR[1] LEQU SOBGTR_OPCODE)
;  1578          4230  2    THEN
;  1579          4231  3            BEGIN
;  1580          4232  3            !++
;  1581          4233  3            ! Handle the second group of substitutions.  THese may be replaced with
;  1582          4234  3            ! the instruction branch, a BRB instruction, and a BRW or JMP
;  1583          4235  3            ! instruction.  This group includes instructions BBSSI through ACBD in
;  1584          4236  3            ! the above table.  The instruction branch must be set to branch around
;  1585          4237  3            ! the BRB instruction.  The BRB instruction must be set to branch around
;  1586          4238  3            ! the BRW instruction.  Therefore, the instruction stream changes from:
;  1587          4239  3            !       <BR INS> TO <X>
;  1588          4240  3            ! TO:
;  1589          4241  3            !       <BR INS> TO .+02        BRB TO .+03     BRW <X>
;  1590          4242  3            !--
;  1591          4243  3            PAT$GB_SUBST_IN[0] = .OLD_INS_PTR[0] + BRB_INS_SIZ + BRW_INS_SIZ; ! Set the stream length
;  1592          4244  3            CH$MOVE(.OLD_INS_PTR[0]-1, CH$PTR(OLD_INS_PTR[1]), CH$PTR(PAT$GB_SUBST_IN[1])); ! Copy old ins strea
;  1593          4245  3            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0] ] = BRB_INS_SIZ;       ! Set displ to br around BRB ins
;  1594          4246  3            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+1 ] = BRB_OPCODE;      ! Set BRB opcode
;  1595          4247  3            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = BRW_INS_SIZ;     ! Set BRB around BRW ins
;  1596          4248  3            PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ] = BRW_OPCODE;      ! Set BRW opcode
;  1597          4249  3            BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
;  1598          4250  3            IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
;  1599          4251  4               (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)                 ! Does displacement fit?
;  1600          4252  3            THEN
;  1601          4253  3                (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0]-1 ])<0,16,1> = .BR_DISPLACEMENT ! Yes, move in displac
;  1602          4254  3            ELSE
;  1603          4255  4                BEGIN
;  1604          4256  4                !++
;  1605          4257  4                ! No, displacement did not fit, therefore use the
;  1606          4258  4                ! second substitution choice.  This includes changing
;  1607          4259  4                ! the BRW to a JMP, and altering the branch around it.
```

```
;  1608          4260  4                   !--
;  1609          4261  4                   PAT$GB_SUBST_IN[0] = .PAT$GB_SUBST_IN[0] + (JMP_INS_SIZ - BRW_INS_SIZ); ! Set a new stream s
;  1610          4262  4                   PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = JMP_INS_SIZ; ! Change BRB displacement around JMP
;  1611          4263  4                   PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ] = JMP_OPCODE; ! Replace the BRW opcode
;  1612          4264  4                   PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+4 ] = PC_DEFERRED; ! Set the instruction mode
;  1613          4265  4                   (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0] + A_BYTE - A_LONGWORD ])<0,32,1> =
;  1614          4266  4                           .BR_DISPLACEMENT - (JMP_INS_SIZ - BRW_INS_SIZ); ! Adjust the displ
;  1615          4267  3                   END;
;  1616          4268  3           END
;  1617          4269  2       ELSE
;  1618          4270  2       !++
;  1619          4271  2       ! The opcode was not one of the second group, therefore check to see if it
;  1620          4272  2       ! was one of the third group.
;  1621          4273  2       !--
;  1622          4274  2       IF (.OLD_INS_PTR[1] EQLU ACBB_OPCODE) OR
;  1623          4275  2          (.OLD_INS_PTR[1] EQLU ACBW_OPCODE) OR
;  1624          4276  2          (.OLD_INS_PTR[1] EQLU ACBL_OPCODE) OR
;  1625          4277  2          (.OLD_INS_PTR[1] EQLU ACBF_OPCODE) OR
;  1626          4278  2          (.OLD_INS_PTR[1] EQLU ACBD_OPCODE) OR
;  1627          4279  2          (.OLD_INS_PTR[1] EQLU %X'FD' AND .OLD_INS_PTR[2] EQLU ACBG_HICODE) OR
;  1628          4280  2          (.OLD_INS_PTR[1] EQLU %X'FD' AND .OLD_INS_PTR[2] EQLU ACBH_HICODE)
;  1629          4281  2       THEN
;  1630          4282  3           BEGIN
;  1631          4283  3           !++
;  1632          4284  3           ! Handle the third group of substitutions.  These may be replaced with
;  1633          4285  3           ! the instruction branch, a BRB instruction, and a JMP instruction.
;  1634          4286  3           ! This group includes instructions ACBB through ACBD in the above table.
;  1635          4287  3           ! The instruction branch must be set to branch around the BRB
;  1636          4288  3           ! instruction.  The BRB instruction must be set to branch around the
;  1637          4289  3           ! BRW instruction.  THerefore, the instruction stream changes from:
;  1638          4290  3           !
;  1639          4291  3           ! CHANGES FROM:
;  1640          4292  3           !       <BR INS> TO <X>
;  1641          4293  3           ! TO:
;  1642          4294  3           !       <BR INS> TO .+02        BRB TO .+06     JMP <X>
;  1643          4295  3           !--
;  1644          4296  3           PAT$GB_SUBST_IN[0] = .OLD_INS_PTR[0] + (JMP_INS_SIZ + BRB_INS_SIZ); ! Set the stream length
;  1645          4297  3           CH$MOVE(.OLD_INS_PTR[0]-2, CH$PTR(OLD_INS_PTR[1]), CH$PTR(PAT$GB_SUBST_IN[1])); ! Copy old ins strea
;  1646          4298  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]-1] = BRB_INS_SIZ;       ! Set displ to br around BRB ins
;  1647          4299  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]] = 0;                   ! Clear other byte of displ word
;  1648          4300  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+1] = BRB_OPCODE;        ! Set BRB opcode
;  1649          4301  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+2] = JMP_INS_SIZ;       ! Set BRB around JMP instruction
;  1650          4302  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+3] = JMP_OPCODE;        ! Set BRW opcode
;  1651          4303  3           PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+4] = PC_DEFERRED;       ! Set instruction mode
;  1652          4304  3           BR_DISPLACEMENT = .PAT$GB_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
;  1653          4305  3           (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0] + A_BYTE - A_LONGWORD ])<0,32,1> = .BR_DISPLACEMENT; ! Adjust
;  1654          4306  3           END
;  1655          4307  2       ELSE
;  1656          4308  2       !++
;  1657          4309  2       ! The opcode was not one of the third group, therefore check to see if it
;  1658          4310  2       ! was one of the fourth group.
;  1659          4311  2       !--
;  1660          4312  2       IF (.OLD_INS_PTR[1] EQL BRB_OPCODE) OR (.OLD_INS_PTR[1] EQL BSBB_OPCODE)
;  1661          4313  2       THEN
;  1662          4314  3           BEGIN
;  1663          4315  3           !++
;  1664          4316  3           ! Handle the fourth group of substitutions.  THese may be replaced with
```

```
;  1665     4317  3          ! the next larger displacement branch instruction of the same type.
;  1666     4318  3          ! This group includes instrucitons BRB and BSBB.  These instructions
;  1667     4319  3          ! can be handled similarly because:
;  1668     4320  3          ! (1) They have the same binary format, and
;  1669     4321  3          ! (2) The difference in opcodes for this branch
;  1670     4322  3          !       displacement and the next larger is the same.
;  1671     4323  3          ! Therefore, because of (1), the variables BRB_INS_SIZ and
;  1672     4324  3          ! BRW_INS_SIZ would be identical to BSBB_INS_SIZ and
;  1673     4325  3          ! BSBW_INS_SIZ.  Also, because of (2), (BRW_OPCODE - BRB_OPCODE)
;  1674     4326  3          ! is the same as (BSBW_OPCODE - BSBB_OPCODE).
;  1675     4327  3          !--
;  1676     4328  3          PAT$GB_SUBST_IN[0] = BRW_INS_SIZ;                       ! Set ins stream size
;  1677     4329  3          PAT$GB_SUBST_IN[1] = .OLD_INS_PTR[1] + (BRW_OPCODE - BRB_OPCODE); ! Set new opcode
;  1678     4330  3          BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new displ
;  1679     4331  3          IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
;  1680     4332  4             (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)              ! Does displ fit?
;  1681     4333  3          THEN
;  1682     4334  3                  (PAT$GB_SUBST_IN[2])<0,16,1> = .BR_DISPLACEMENT ! Yes, move displ into stream
;  1683     4335  3          ELSE
;  1684     4336  4                  BEGIN
;  1685     4337  4                  !++
;  1686     4338  4                  ! No, displacement did not fit.  A longword displacement must be
;  1687     4339  4                  ! used.  Therefore, convert to a JSB or JMP instruction.
;  1688     4340  4                  !--
;  1689     4341  4                  PAT$GB_SUBST_IN[0] = JMP_INS_SIZ;           ! Set new stream size
;  1690     4342  4                  PAT$GB_SUBST_IN[1] = .PAT$GB_SUBST_IN[1] + (JMP_OPCODE - BRW_OPCODE); ! Set new opcode
;  1691     4343  4                  PAT$GB_SUBST_IN[2] = PC_DEFERRED;           ! Set instruction mode
;  1692     4344  4                  (PAT$GB_SUBST_IN[3])<0,32,1> = .BR_DISPLACEMENT -
;  1693     4345  4                          (JMP_INS_SIZ - BRW_INS_SIZ);        ! Compute new displacement
;  1694     4346  4                  END
;  1695     4347  3          END
;  1696     4348  2 ELSE
;  1697     4349  2 !++
;  1698     4350  2 ! The opcode was not one of the fourth group, therefore check to see if it
;  1699     4351  2 ! was one of the fifth group.
;  1700     4352  2 !--
;  1701     4353  3 IF (.OLD_INS_PTR[1] EQL BRW_OPCODE) OR (.OLD_INS_PTR[1] EQL BSBW_OPCODE)
;  1702     4354  2 THEN
;  1703     4355  3          BEGIN
;  1704     4356  3          !++
;  1705     4357  3          ! Handle the fifth group of substitutions.  These may be replaced with
;  1706     4358  3          ! the next larger displacement branch instruction of the same type.
;  1707     4359  3          ! This group includes instructions BRW and BSBW.  These instructions
;  1708     4360  3          ! can be handled similarly because:
;  1709     4361  3          ! (1) They have the same binary format, and
;  1710     4362  3          ! (2) The difference in opcodes for this branch displacement
;  1711     4363  3          !       and the next larger is the same.
;  1712     4364  3          ! Therefore, beacuse of (1), the variables JSB_INS_SIZ and JMP_INS_SIZ
;  1713     4365  3          ! would be identical.  Also, because of (2), (JMP_OPCODE - BRW_OPCODE)
;  1714     4366  3          ! is the same as (JSB_OPCODE - BSBW_OPCODE).
;  1715     4367  3          !--
;  1716     4368  3          PAT$GB_SUBST_IN[0] = JMP_INS_SIZ;                       ! Set ins stream size
;  1717     4369  3          PAT$GB_SUBST_IN[1] = .OLD_INS_PTR[1] + (JMP_OPCODE - BRW_OPCODE); ! Set opcode
;  1718     4370  3          PAT$GB_SUBST_IN[2] = PC_DEFERRED;                       ! Set ins mode
;  1719     4371  3          BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Get displacement
;  1720     4372  3          (PAT$GB_SUBST_IN[3])<0,32,1> = .BR_DISPLACEMENT;        ! Compute br displ
;  1721     4373  3          END
```

```
;  1722          4374  2 ELSE
;  1723          4375  2         RETURN (FALSE);
;  1724          4376  2
;  1725          4377  2 RETURN (TRUE);
;  1726          4378  1 END;                                                    ! End of PAT$SUBST_INS


                                      0FFC 00000          .ENTRY  PAT$SUBST_INS, Save R2,R3,R4,R5,R6,R7,R8,-      ; 4032
                                                                  R9,R10,R11
                              5B 00000000G   EF   9E 00002      MOVAB   PAT$GL_BR_DISPL, R11
                              5A 00000000G   EF   9E 00009      MOVAB   PAT$GB_SUBST_IN, R10
                              59       04    AC   D0 00010      MOVL    OLD_INS_PTR, R9                               ; 4185
                              57       01    A9   9A 00014      MOVZBL  1(R9), R7
                              12             57   91 00018      CMPB    R7, #18
                                             05   1F 0001B      BLSSU   1$
                              15             57   91 0001D      CMPB    R7, #21
                                             22   1B 00020      BLEQU   4$
                              18             57   91 00022 1$:  CMPB    R7, #24                                       ; 4186
                                             05   1F 00025      BLSSU   2$
                              1F             57   91 00027      CMPB    R7, #31
                                             18   1B 0002A      BLEQU   4$
                          E0  8F             57   91 0002C 2$:  CMPB    R7, #224                                      ; 4187
                                             06   1F 00030      BLSSU   3$
                          E5  8F             57   91 00032      CMPB    R7, #229
                                             0C   1B 00036      BLEQU   4$
                          E8  8F             57   91 00038 3$:  CMPB    R7, #232                                      ; 4188
                                             6B   1F 0003C      BLSSU   8$
                          E9  8F             57   91 0003E      CMPB    R7, #233
                                             65   1A 00042      BGTRU   8$
                              56             69   9A 00044 4$:  MOVZBL  (R9), R6                                     ; 4195
                      6A                     56   03   81 00047      ADDB3   #3, R6, PAT$GB_SUBST_IN
                                             06   57   E9 0004B      BLBC    R7, 5$                                   ; 4196
                              50         FF  A7   9E 0004E      MOVAB   -1(R7), R0
                                             04   11 00052      BRB     6$
                              50         01  A7   9E 00054 5$:  MOVAB   1(R7), R0                                    ; 4197
                          01  AA             50   90 00058 6$:  MOVB    R0, PAT$GB_SUBST_IN+1                        ; 4196
                              50         FE  A6   9E 0005C      MOVAB   -2(R6), R0                                   ; 4198
                  02  AA      02        A9   50   28 00060      MOVC3   R0, 2(R9), PAT$GB_SUBST_IN+2
                          6A46                   03   90 00066      MOVB    #3, PAT$GB_SUBST_IN[R6]                      ; 4199
                          01  AA46              31   90 0006A      MOVB    #49, PAT$GB_SUBST_IN+1[R6]                   ; 4200
                      50                     6B   56   C1 0006F      ADDL3   R6, PAT$GL_BR_DISPL, R0                      ; 4201
                      58                     50   6A   9A 00073      MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT
                      58                     50   58   C3 00076      SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT
                          00007FFF           8F   58   D1 0007A      CMPL    BR_DISPLACEMENT, #32767                      ; 4202
                                                  0F   14 00081      BGTR    7$
                          FFFF8000           8F   58   D1 00083      CMPL    BR_DISPLACEMENT, #-32768                     ; 4203
                                                  06   19 0008A      BLSS    7$
                                         02 AA46  9F   0008C      PUSHAB  PAT$GB_SUBST_IN+2[R6]                        ; 4205
                                                  77   11 00090      BRB     10$
                                      6A          03   80 00092 7$:  ADDB2   #3, PAT$GB_SUBST_IN                       ; 4213
                                  6A46            06   90 00095      MOVB    #6, PAT$GB_SUBST_IN[R6]                  ; 4214
                              01  AA46            17   90 00099      MOVB    #23, PAT$GB_SUBST_IN+1[R6]               ; 4215
                              02  AA46            11   8E 0009E      MNEGB   #17, PAT$GB_SUBST_IN+2[R6]               ; 4216
                                         03 AA46  9F   000A3      PUSHAB  PAT$GB_SUBST_IN+3[R6]                        ; 4217
```

```
                                                7E  11 000A7          BRB     12$                                                     ;
                                    E6   8F     57  91 000A9 8$:      CMPB    R7, #230                                        4227    ;
                                                12  13 000AD         BEQL    9$                                                      ;
                                    E7   8F     57  91 000AF         CMPB    R7, #231                                        4228    ;
                                                0C  13 000B3         BEQL    9$                                                      ;
                                    F2   8F     57  91 000B5         CMPB    R7, #242                                        4229    ;
                                                72  1F 000B9         BLSSU   14$                                                     ;
                                    F5   8F     57  91 000BB         CMPB    R7, #245                                                ;
                                                6C  1A 000BF         BGTRU   14$                                                     ;
                                         56     69  9A 000C1 9$:      MOVZBL  (R9), R6                                        4243    ;
                                6A       56     05  81 000C4         ADDB3   #5, R6, PAT$GB_SUBST_IN                                 ;
                                         50  FF A6  9E 000C8         MOVAB   -1(R6), R0                                      4244    ;
                            01  AA   01   A9     50  28 000CC         MOVC3   R0, 1(R9), PAT$GB_SUBST_IN+1                            ;
                                   6A46      02  90 000D2         MOVB    #2, PAT$GB_SUBST_IN[R6]                         4245    ;
                                   01 AA46   11  90 000D6         MOVB    #17, PAT$GB_SUBST_IN+1[R6]                      4246    ;
                                   02 AA46   03  90 000DB         MOVB    #3, PAT$GB_SUBST_IN+2[R6]                       4247    ;
                                   03 AA46   31  90 000E0         MOVB    #49, PAT$GB_SUBST_IN+3[R6]                      4248    ;
                                50     6B     56  C1 000E5         ADDL3   R6, PAT$GL_BR_DISPL, R0                         4249    ;
                                58     58     6A  9A 000E9         MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT                        ;
                                50     50     58  C3 000EC         SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT                    ;
                                   00007FFF   8F  58  D1 000F0         CMPL    BR_DISPLACEMENT, #32767                         4250    ;
                                                15  14 000F7         BGTR    11$                                                     ;
                                   FFFF8000   8F  58  D1 000F9         CMPL    BR_DISPLACEMENT, #-32768                        4251    ;
                                                0C  19 00100         BLSS    11$                                                     ;
                                         50     6A  9A 00102         MOVZBL  PAT$GB_SUBST_IN, R0                             4253    ;
                                             FF AA40 9F 00105         PUSHAB  PAT$GB_SUBST_IN-1[R0]                                   ;
                                         9E     58  B0 00109 10$:      MOVW    BR_DISPLACEMENT, @(SP)+                                 ;
                                                1D  11 0010C         BRB     13$                                                     ;
                                         6A     03  80 0010E 11$:      ADDB2   #3, PAT$GB_SUBST_IN                             4261    ;
                                   02 AA46      06  90 00111         MOVB    #6, PAT$GB_SUBST_IN+2[R6]                       4262    ;
                                   03 AA46      17  90 00116         MOVB    #23, PAT$GB_SUBST_IN+3[R6]                      4263    ;
                                   04 AA46      11  8E 0011B         MNEGB   #17, PAT$GB_SUBST_IN+4[R6]                      4264    ;
                                         50     6A  9A 00120         MOVZBL  PAT$GB_SUBST_IN, R0                             4265    ;
                                             FD AA40 9F 00123         PUSHAB  PAT$GB_SUBST_IN-3[R0]                           4266    ;
                                         9E  FD  A8  9E 00127 12$:      MOVAB   -3(R8), @(SP)+                                  4227    ;
                                                79  11 0012B 13$:      BRB     17$                                             4274    ;
                                    9D   8F     57  91 0012D 14$:      CMPB    R7, #157                                                ;
                                                31  13 00131         BEQL    16$                                                     ;
                                    3D          57  91 00133         CMPB    R7, #61                                         4275    ;
                                                2C  13 00136         BEQL    16$                                                     ;
                                    F1   8F     57  91 00138         CMPB    R7, #241                                        4276    ;
                                                26  13 0013C         BEQL    16$                                                     ;
                                    4F   8F     57  91 0013E         CMPB    R7, #79                                         4277    ;
                                                20  13 00142         BEQL    16$                                                     ;
                                    6F   8F     57  91 00144         CMPB    R7, #111                                        4278    ;
                                                1A  13 00148         BEQL    16$                                                     ;
                                    FD   8F     57  91 0014A         CMPB    R7, #253                                        4279    ;
                                                07  12 0014E         BNEQ    15$                                                     ;
                                    4F   8F  02 A9  91 00150         CMPB    2(R9), #79                                              ;
                                                0D  13 00155         BEQL    16$                                                     ;
                                    FD   8F     57  91 00157 15$:      CMPB    R7, #253                                        4280    ;
                                                4B  12 0015B         BNEQ    18$                                                     ;
                                    6F   8F  02 A9  91 0015D         CMPB    2(R9), #111                                             ;
                                                44  12 00162         BNEQ    18$                                                     ;
                                         56     69  9A 00164 16$:      MOVZBL  (R9), R6                                        4296    ;
                                6A       56     08  81 00167         ADDB3   #8, R6, PAT$GB_SUBST_IN                                 ;
                                         50  FE A6  9E 0016B         MOVAB   -2(R6), R0                                      4297    ;
```

```
      01   AA        01   A9         50  28 0016F          MOVC3   R0, 1(R9), PAT$GB_SUBST_IN+1            ;
                     FF AA46         02  90 00175          MOVB    #2, PAT$GB_SUBST_IN-1[R6]               ; 4298
                                   6A46  94 0017A          CLRB    PAT$GB_SUBST_IN[R6]                     ; 4299
                     01 AA46         11  90 0017D          MOVB    #17, PAT$GB_SUBST_IN+1[R6]              ; 4300
                     02 AA46         06  90 00182          MOVB    #6, PAT$GB_SUBST_IN+2[R6]               ; 4301
                     03 AA46         17  90 00187          MOVB    #23, PAT$GB_SUBST_IN+3[R6]              ; 4302
                     04 AA46         11  8E 0018C          MNEGB   #17, PAT$GB_SUBST_IN+4[R6]              ; 4303
           50             6B         56  C1 00191          ADDL3   R6, PAT$GL_BR_DISPL, R0                 ; 4304
                          58         6A  9A 00195          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT        ;
           58             50         58  C3 00198          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT    ;
                          50         6A  9A 0019C          MOVZBL  PAT$GB_SUBST_IN, R0                     ; 4305
                                FD AA40  9F 0019F          PUSHAB  PAT$GB_SUBST_IN-3[R0]                   ;
                          9E         58  D0 001A3          MOVL    BR_DISPLACEMENT, @(SP)+                 ;
                                     70  11 001A6  17$:    BRB     23$                                     ; 4274
                          11         57  91 001A8  18$:    CMPB    R7, #17                                 ; 4312
                                     05  13 001AB          BEQL    19$                                     ;
                          10         57  91 001AD          CMPB    R7, #16                                 ;
                                     3F  12 001B0          BNEQ    21$                                     ;
                          6A         03  90 001B2  19$:    MOVB    #3, PAT$GB_SUBST_IN                     ; 4328
      01   AA             57         20  81 001B5          ADDB3   #32, R7, PAT$GB_SUBST_IN+1              ; 4329
                          50         69  9A 001BA          MOVZBL  (R9), R0                                ; 4330
                          50         6B  C0 001BD          ADDL2   PAT$GL_BR_DISPL, R0                     ;
                          58         6A  9A 001C0          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT        ;
           58             50         58  C3 001C3          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT    ;
              00007FFF    8F         58  D1 001C7          CMPL    BR_DISPLACEMENT, #32767                 ; 4331
                                     0F  14 001CE          BGTR    20$                                     ;
              FFFF8000    8F         58  D1 001D0          CMPL    BR_DISPLACEMENT, #-32768                ; 4332
                                     06  19 001D7          BLSS    20$                                     ;
                     02   AA         58  B0 001D9          MOVW    BR_DISPLACEMENT, PAT$GB_SUBST_IN+2      ; 4334
                                     39  11 001DD          BRB     23$                                     ;
                          6A         06  90 001DF  20$:    MOVB    #6, PAT$GB_SUBST_IN                     ; 4341
                     01   AA         1A  82 001E2          SUBB2   #26, PAT$GB_SUBST_IN+1                  ; 4342
                     02   AA         11  8E 001E6          MNEGB   #17, PAT$GB_SUBST_IN+2                  ; 4343
                     03   AA    FD   A8  9E 001EA          MOVAB   -3(R8), PAT$GB_SUBST_IN+3               ; 4344
                                     27  11 001EF          BRB     23$                                     ; 4331
                          31         57  91 001F1  21$:    CMPB    R7, #49                                 ; 4353
                                     05  13 001F4          BEQL    22$                                     ;
                          30         57  91 001F6          CMPB    R7, #48                                 ;
                                     21  12 001F9          BNEQ    24$                                     ;
                          6A         06  90 001FB  22$:    MOVB    #6, PAT$GB_SUBST_IN                     ; 4368
      01   AA             57         1A  83 001FE          SUBB3   #26, R7, PAT$GB_SUBST_IN+1              ; 4369
                     02   AA         11  8E 00203          MNEGB   #17, PAT$GB_SUBST_IN+2                  ; 4370
                          50         69  9A 00207          MOVZBL  (R9), R0                                ; 4371
                          50         6B  C0 0020A          ADDL2   PAT$GL_BR_DISPL, R0                     ;
                          58         6A  9A 0020D          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT        ;
           58             50         58  C3 00210          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT    ;
                     03   AA         58  D0 00214          MOVL    BR_DISPLACEMENT, PAT$GB_SUBST_IN+3      ; 4372
                          50         01  D0 00218  23$:    MOVL    #1, R0                                  ; 4377
                                         04 0021B          RET                                             ;
                                     50  D4 0021C  24$:    CLRL    R0                                      ; 4378
                                         04 0021E          RET                                             ;

; Routine Size:  543 bytes,    Routine Base:  _PAT$CODE + 0B5B
```

```
;1728        4379  1 GLOBAL ROUTINE PAT$OUT_MEM_LOC (LOCATION, PREFIX_STG, ASM_DIR_TBL, CASE_TBL) =
;1729        4380  1
;1730        4381  1 !++
;1731        4382  1 ! FUNCTIONAL DESCRIPTION:
;1732        4383  1 !
;1733        4384  1 !       Outputs the value of a memory location to the output
;1734        4385  1 !       device. If this routine is called as a result of an EXAMINE
;1735        4386  1 !       command, the location itself is also displayed, followed by
;1736        4387  1 !       a colon and a tab.
;1737        4388  1 !
;1738        4389  1 !       The appropriate mode settings are used to control the output
;1739        4390  1 !       style.
;1740        4391  1 !
;1741        4392  1 ! CALLING SEQUENCE:
;1742        4393  1 !
;1743        4394  1 !       PAT$OUT_MEM_LOC ()
;1744        4395  1 !
;1745        4396  1 ! INPUTS:
;1746        4397  1 !
;1747        4398  1 !       LOCATION        - Unmapped location whose contents are to be displayed.
;1748        4399  1 !       PREFIX_STG      - Prefix string to output before the location
;1749        4400  1 !                         0 = NONE
;1750        4401  1 !       ASM_DIR_TBL     - Address of assembler directive table descriptor
;1751        4402  1 !       CASE_TBL        - TRUE => Print CASE dispatch tables
;1752        4403  1 !
;1753        4404  1 ! IMPLICIT INPUTS:
;1754        4405  1 !
;1755        4406  1 !       PAT$GL_CONTEXT [EXAMINE_BIT]    - If this bit is set, the address of the
;1756        4407  1 !                                         value is also displayed.
;1757        4408  1 !       PAT$GL_MOD_PTR                  - Pointer to the current mode level
;1758        4409  1 !
;1759        4410  1 ! OUTPUTS:
;1760        4411  1 !
;1761        4412  1 !       TRUE for success, FALSE for failure.
;1762        4413  1 !
;1763        4414  1 ! IMPLICIT OUTPUTS:
;1764        4415  1 !
;1765        4416  1 !       NONE
;1766        4417  1 !
;1767        4418  1 ! ROUTINE VALUE:
;1768        4419  1 !
;1769        4420  1 !       TRUE or FALSE
;1770        4421  1 !
;1771        4422  1 ! SIDE EFFECTS:
;1772        4423  1 !
;1773        4424  1 !       Data is output to the data device.  An error message is produced if the
;1774        4425  1 !       memory location is not readable.
;1775        4426  1 !
;1776        4427  1 !--
;1777        4428  1
;1778        4429  2 BEGIN
;1779        4430  2 LOCAL
;1780        4431  2         MAPPED_LOC : REF VECTOR[,BYTE],                 ! Mapped address of deposit location
;1781        4432  2         ISE_ADDR: REF VECTOR[,BYTE],                    ! ISE address for deposit location
;1782        4433  2         OUT_VALUES : VECTOR[ TTY_OUT_WIDTH, BYTE],
;1783        4434  2         OUTPUT_BUFFER : VECTOR[ TTY_OUT_WIDTH, BYTE];
;1784        4435  2
```

```
; 1785        4436  2 !++
; 1786        4437  2 ! Initialize buffer address and size.
; 1787        4438  2 !--
; 1788        4439  2 PAT$CP_OUT_STR = OUTPUT_BUFFER + 1;
; 1789        4440  2 PAT$GL_BUF_SIZ = 0;
; 1790        4441  2
; 1791        4442  2 !++
; 1792        4443  2 ! First check if there is a prefix string to be output.
; 1793        4444  2 !--
; 1794        4445  3 IF (.PREFIX_STG NEQ 0)
; 1795        4446  2 THEN
; 1796        4447  2         PAT$FAO_PUT(.PREFIX_STG);
; 1797        4448  2
; 1798        4449  2 !++
; 1799        4450  2 ! Now if the examine bit is set then output a location which is mapped
; 1800        4451  2 ! by PATCH.  If the examine bit is not set, then output an expression
; 1801        4452  2 ! for the EVALUATE command.
; 1802        4453  2 !--
; 1803        4454  2 IF .PAT$GL_CONTEXT [EXAMINE_BIT]
; 1804        4455  2 THEN
; 1805        4456  3         BEGIN
; 1806        4457  3         !++
; 1807        4458  3         ! Print the address, making it come out as a longword regardless of
; 1808        4459  3         ! the current output mode length.
; 1809        4460  3         !--
; 1810        4461  3         PAT$MAP_ADDR(.LOCATION,MAPPED_LOC,ISE_ADDR);            ! Compute mapped address
; 1811        4462  3         PAT$OUT_SYM_VAL(.LOCATION, LONG_LENGTH, 0);
; 1812        4463  3         PAT$GL_LAST_LOC = .LOCATION;
; 1813        4464  3         PAT$GB_LOC_TYPE = MEMORY_LOC;
; 1814        4465  3         PAT$FAO_PUT ( COLON_TAB_STG );
; 1815        4466  3
; 1816        4467  3         !++
; 1817        4468  3         ! Handle output as symbolic instructions separately.
; 1818        4469  3         !--
; 1819        4470  4         IF( .PAT$GB_MOD_PTR[ MODE_INSTRUC ] )
; 1820        4471  3         THEN
; 1821        4472  4                 IF ((LOCATION = PAT$INS_DECODE (.LOCATION, OUTPUT_BUFFER, LOCATION, .ASM_DIR_TBL, .CASE_TBL)
; 1822        4473  3                 THEN
; 1823        4474  4                         BEGIN
; 1824        4475  4                         SIGNAL (PAT$_NODECODE);
; 1825        4476  4                         RETURN(FALSE);
; 1826        4477  4                         END
; 1827        4478  3                 ELSE
; 1828        4479  4                         BEGIN
; 1829        4480  4                         PAT$MAP_ADDR (.LOCATION, MAPPED_LOC, ISE_ADDR);
; 1830        4481  4                         IF .PAT$GL_CONTEXT [EXAMINE_BIT]
; 1831        4482  4                         THEN
; 1832        4483  4                                 PAT$GL_NEXT_LOC = .LOCATION;
; 1833        4484  4
; 1834        4485  4                         !++
; 1835        4486  4                         ! PAT$GL_LAST_VAL may be set within PAT$INS_DECODE.
; 1836        4487  4                         !--
; 1837        4488  4                         END
; 1838        4489  3         ELSE
; 1839        4490  4                 BEGIN
; 1840        4491  4                 !++
; 1841        4492  4                 ! Special attention for ascii output.
```

```
;  1842     4493  4                !--
;  1843     4494  5                IF (.PAT$GB_MOD_PTR [MODE_ASCII])
;  1844     4495  4                THEN
;  1845     4496  4                        !++
;  1846     4497  4                        ! Simply output the number of characters
;  1847     4498  4                        ! implied by the current MODE_LENGTH setting.
;  1848     4499  4                        !--
;  1849     4500  5                        BEGIN
;  1850     4501  5                        PAT$GET_VALUE (.LOCATION, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
;  1851     4502  5                        PAT$FAO_PUT (CS_ASCII, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
;  1852     4503  5                        PAT$GL_NEXT_LOC = .LOCATION + .PAT$GB_MOD_PTR [MODE_LENGTH];
;  1853     4504  5                        PAT$GL_LAST_VAL = .(.MAPPED_LOC) <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
;  1854     4505  5                        END
;  1855     4506  4                ELSE                                    ! Otherwise we have one of the usual modes
;  1856     4507  4                        IF .PAT$GL_CONTEXT [EXAMINE_BIT]
;  1857     4508  4                        THEN
;  1858     4509  5                                BEGIN
;  1859     4510  5                                PAT$GET_VALUE(.LOCATION, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
;  1860     4511  5                                PAT$OUT_NUM_VAL(.OUT_VALUES, 0, 0, TRUE);
;  1861     4512  5                                PAT$GL_NEXT_LOC = .LOCATION + .PAT$GB_MOD_PTR [MODE_LENGTH];
;  1862     4513  5                                PAT$GL_LAST_VAL = .OUT_VALUES <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
;  1863     4514  4                                END;
;  1864     4515  4                END
;  1865     4516  3        END
;  1866     4517  2 ELSE
;  1867     4518  3        BEGIN
;  1868     4519  3        !++
;  1869     4520  3        ! Output the value for the EVALUATE command here then return.
;  1870     4521  3        ! All other commands have set the examine bit.  Check for different
;  1871     4522  3        ! output modes, literal or instruction.
;  1872     4523  3        !--
;  1873     4524  4        IF (.PAT$GL_CONTEXT[LITERAL_BIT])
;  1874     4525  3        THEN
;  1875     4526  4                BEGIN
;  1876     4527  4                !++
;  1877     4528  4                ! Call a routine which does the whole thing - including
;  1878     4529  4                ! flushing the output and producing an error message if no
;  1879     4530  4                ! such literal translation can be found.
;  1880     4531  4                !--
;  1881     4532  4                DISPLAY_LVTS(..LOCATION);
;  1882     4533  4
;  1883     4534  4                !++
;  1884     4535  4                ! If the above routine returns then at least one literal
;  1885     4536  4                ! translation was found.  This form of evaluate sets the
;  1886     4537  4                ! psuedo '\' (last value displayed) only.
;  1887     4538  4                !--
;  1888     4539  4                PAT$GL_LAST_VAL = ..LOCATION;
;  1889     4540  4                RETURN(TRUE);
;  1890     4541  3                END;
;  1891     4542  3
;  1892     4543  3        !++
;  1893     4544  3        ! Instruction mode works only if /LITERAL was not specified.
;  1894     4545  3        !--
;  1895     4546  4        IF (.PAT$GB_MOD_PTR[MODE_INSTRUC])
;  1896     4547  3        THEN
;  1897     4548  4                BEGIN
;  1898     4549  4                LOCAL
```

```
;  1899          4550  4                       COUNT,
;  1900          4551  4                       ENCODED_BUF : VECTOR[38,BYTE];
;  1901          4552  5                   IF (NOT PAT$INS_ENCODE(..LOCATION, ENCODED_BUF, 0))
;  1902          4553  4                   THEN
;  1903          4554  4                       SIGNAL(PAT$_NOENCODE, 1, ..LOCATION);
;  1904          4555  4                   COUNT = .ENCODED_BUF[0];
;  1905          4556  4                   DO
;  1906          4557  5                       BEGIN
;  1907          4558  5                       PAT$OUT_NUM_VAL(.ENCODED_BUF[.COUNT], BYTE_LENGTH, HEX_RADIX, FALSE);
;  1908          4559  5                       COUNT = .COUNT - 1;
;  1909          4560  5                       END
;  1910          4561  4                   UNTIL .COUNT EQL 0;
;  1911          4562  4                   END
;  1912          4563  3           ELSE
;  1913          4564  4                   BEGIN
;  1914          4565  4                   PAT$OUT_NUM_VAL(..LOCATION, 0, 0, TRUE);
;  1915          4566  4                   PAT$GL_LAST_VAL = .(LOCATION) <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
;  1916          4567  3                   END;
;  1917          4568  2           END;
;  1918          4569  2
;  1919          4570  2 !++
;  1920          4571  2 ! Write out the string and return.
;  1921          4572  2 !--
;  1922          4573  2 PAT$OUT_PUT( OUTPUT_BUFFER );
;  1923          4574  2
;  1924          4575  2 RETURN TRUE
;  1925          4576  1 END;


                                OFFC 00000          .ENTRY  PAT$OUT_MEM_LOC, Save R2,R3,R4,R5,R6,R7,R8,-; 4379
                                                            R9,R10,R11
                      5B 00000000G   00  9E 00002   MOVAB   LIB$SIGNAL, R11
                      5A 00000000G   EF  9E 00009   MOVAB   PAT$MAP_ADDR, R10
                      59 00000000G   EF  9E 00010   MOVAB   PAT$OUT_NUM_VAL, R9
                      58 00000000G   EF  9E 00017   MOVAB   PAT$GL_NEXT_LOC, R8
                      57 00000000G   EF  9E 0001E   MOVAB   PAT$FAO_PUT, R7
                      56 00000000G   EF  9E 00025   MOVAB   PAT$GL_LAST_VAL, R6
                      55 00000000G   EF  9E 0002C   MOVAB   PAT$GL_CONTEXT, R5
                      54 00000000G   EF  9E 00033   MOVAB   PAT$GB_MOD_PTR, R4
                      5E      FEC8   CE  9E 0003A   MOVAB   -312(SP), SP
00000000G             EF        31   AE  9E 0003F   MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR                4439
                            00000000G  EF  D4 00047   CLRL    PAT$GL_BUF_SIZ                                 4440
                                  08   AC  D5 0004D   TSTL    PREFIX_STG                                     4445
                                       06  13 00050   BEQL    1$
                                  08   AC  DD 00052   PUSHL   PREFIX_STG                                    4447
                            67              01  FB 00055   CALLS   #1, PAT$FAO_PUT
                            53        04    AC  D0 00058 1$:  MOVL    LOCATION, R3                           4461
                            03        01    A5  E8 0005C   BLBS    PAT$GL_CONTEXT+1, 2$                   4454
                                    00EB    31  00060   BRW     7$
                                       5E   DD  00063 2$:  PUSHL   SP                                      4461
                                  08   AE  9F  00065   PUSHAB  MAPPED_LOC
                                       53  DD  00068   PUSHL   R3
                            6A                03  FB 0006A   CALLS   #3, PAT$MAP_ADDR
                            7E                04  7D 0006D   MOVQ    #4, -(SP)                              4462
```

```
                                53  DD 00070          PUSHL   R3                                                      ;
00000000G  EF                   03  FB 00072          CALLS   #3, PAT$OUT_SYM_VAL                                     ;
00000000G  EF                   53  D0 00079          MOVL    R3, PAT$GL_LAST_LOC                                     ; 4463
                00000000G       EF  94 00080          CLRB    PAT$GB_LOC_TYPE                                         ; 4464
                00000000'       EF  9F 00086          PUSHAB  COLON_TAB_STG                                           ; 4465
                67              01  FB 0008C          CALLS   #1, PAT$FAO_PUT                                         ;
                52              64  D0 0008F          MOVL    PAT$GB_MOD_PTR, R2                                      ; 4470
                3A          03  A2  E9 00092          BLBC    3(R2), 4$                                               ;
                7E          0C  AC  7D 00096          MOVQ    ASM_DIR_TBL, -(SP)                                      ; 4472
                            04  AC  9F 0009A          PUSHAB  LOCATION                                                ;
                            3C  AE  9F 0009D          PUSHAB  OUTPUT_BUFFER                                           ;
                                53  DD 000A0          PUSHL   R3                                                      ;
00000000G  EF                   05  FB 000A2          CALLS   #5, PAT$INS_DECODE                                      ;
        04  AC                  50  D0 000A9          MOVL    R0, LOCATION                                            ;
                                0C  12 000AD          BNEQ    3$                                                      ;
                006D8102        8F  DD 000AF          PUSHL   #7176450                                                ; 4475
                6B                  01  FB 000B5      CALLS   #1, LIB$SIGNAL                                          ;
                            010B    31  000B8          BRW     14$                                                     ; 4476
                                5E  DD 000BB  3$:     PUSHL   SP                                                      ; 4480
                            08  AE  9F 000BD          PUSHAB  MAPPED_LOC                                              ;
                            04  AC  DD 000C0          PUSHL   LOCATION                                                ;
                6A                  03  FB 000C3      CALLS   #3, PAT$MAP_ADDR                                        ;
                49          01  A5  E9 000C6          BLBC    PAT$GL_CONTEXT+1, 5$                                    ; 4481
                68          04  AC  D0 000CA          MOVL    LOCATION, PAT$GL_NEXT_LOC                               ; 4483
                                7C  11 000CE          BRB     6$                                                      ; 4472
                3F          04  A2  E9 000D0  4$:     BLBC    4(R2), 5$                                               ; 4494
                        FF7C    CD  9F 000D4          PUSHAB  OUT_VALUES                                              ; 4501
                7E          01  A2  9A 000D8          MOVZBL  1(R2), -(SP)                                            ;
                                53  DD 000DC          PUSHL   R3                                                      ;
00000000G  EF                   03  FB 000DE          CALLS   #3, PAT$GET_VALUE                                       ;
                        FF7C    CD  9F 000E5          PUSHAB  OUT_VALUES                                              ; 4502
                50              64  D0 000E9          MOVL    PAT$GB_MOD_PTR, R0                                      ;
                7E          01  A0  9A 000EC          MOVZBL  1(R0), -(SP)                                            ;
                    00000000'   EF  9F 000F0          PUSHAB  CS_ASCII                                                ;
                67              03  FB 000F6          CALLS   #3, PAT$FAO_PUT                                         ;
                50              64  D0 000F9          MOVL    PAT$GB_MOD_PTR, R0                                      ; 4503
                51          01  A0  9A 000FC          MOVZBL  1(R0), R1                                               ;
            68                  53  C1 00100          ADDL3   R1, R3, PAT$GL_NEXT_LOC                                 ;
                50          01  A0  9A 00104          MOVZBL  1(R0), R0                                               ; 4504
                50              08  C4 00108          MULL2   #8, R0                                                  ;
66      04  BE                  50  00  EF 0010B      EXTZV   #0, R0, @MAPPED_LOC, PAT$GL_LAST_VAL                    ;
                                39  11 00111          BRB     6$                                                      ; 4494
                35          01  A5  E9 00113  5$:     BLBC    PAT$GL_CONTEXT+1, 6$                                    ; 4507
                        FF7C    CD  9F 00117          PUSHAB  OUT_VALUES                                              ; 4510
                7E          01  A2  9A 0011B          MOVZBL  1(R2), -(SP)                                            ;
                                53  DD 0011F          PUSHL   R3                                                      ;
            00000000G           EF  03  FB 00121      CALLS   #3, PAT$GET_VALUE                                       ;
                                01  DD 00128          PUSHL   #1                                                      ; 4511
                                7E  7C 0012A          CLRQ    -(SP)                                                   ;
                        FF7C    CD  DD 0012C          PUSHL   OUT_VALUES                                              ;
                69                  04  FB 00130      CALLS   #4, PAT$OUT_NUM_VAL                                     ;
                50                  64  D0 00133      MOVL    PAT$GB_MOD_PTR, R0                                      ; 4512
                51          01  A0  9A 00136          MOVZBL  1(R0), R1                                               ;
            68                  53  C1 0013A          ADDL3   R1, R3, PAT$GL_NEXT_LOC                                 ;
                50          01  A0  9A 0013E          MOVZBL  1(R0), R0                                               ; 4513
                50              08  C4 00142          MULL2   #8, R0                                                  ;
66      FF7C    CD              50  00  EF 00145      EXTZV   #0, R0, OUT_VALUES, PAT$GL_LAST_VAL                      ;
```

```
                                        6A  11 0014C 6$:     BRB     12$                                             ;4470
                10      03  A5          01  E1 0014E 7$:     BBC     #1, PAT$GL_CONTEXT+3, 8$                        ;4524
                                    04  BC  DD 00153         PUSHL   @LOCATION                                       ;4532
                    00000000V   EF      01  FB 00156         CALLS   #1, DISPLAY_LVTS                                ;
                                66      04  BC  D0 00159     MOVL    @LOCATION, PAT$GL_LAST_VAL                      ;4539
                                    5F  11  00161             BRB     13$                                             ;4540
                                50          64  D0 00163 8$:  MOVL    PAT$GB_MOD_PTR, R0                              ;4546
                                35      03  A0  E9 00166     BLBC    3(R0), 11$                                      ;
                                        7E  D4 0016A         CLRL    -(SP)                                           ;4552
                                    0C  AE  9F 0016C         PUSHAB  ENCODED_BUF                                     ;
                                        63  DD 0016F         PUSHL   (R3)                                            ;
                    00000000G   EF      03  FB 00171         CALLS   #3, PAT$INS_ENCODE                              ;
                                0D          50  E8 00178     BLBS    R0, 9$                                          ;
                                        63  DD 0017B         PUSHL   (R3)                                            ;4554
                                        01  DD 0017D         PUSHL   #1                                              ;
                                006D810A    8F  DD 0017F     PUSHL   #7176458                                        ;
                                6B          03  FB 00185     CALLS   #3, LIB$SIGNAL                                  ;
                                52      08  AE  9A 00188 9$:  MOVZBL  ENCODED_BUF, COUNT                             ;4555
                                7E          10  7D 0018C 10$:  MOVQ    #16, -(SP)                                  ;4558
                                        01  DD 0018F         PUSHL   #1                                              ;
                                7E      14 AE42 9A 00191     MOVZBL  ENCODED_BUF[COUNT], -(SP)                       ;
                                69          04  FB 00196     CALLS   #4, PAT$OUT_NUM_VAL                             ;
                                        52  D7 00199         DECL    COUNT                                           ;4559
                                        EF  12 0019B         BNEQ    10$                                             ;4561
                                        19  11 0019D         BRB     12$                                             ;4546
                                        01  DD 0019F 11$:     PUSHL   #1                                              ;4565
                                        7E  7C 001A1         CLRQ    -(SP)                                           ;
                                        63  DD 001A3         PUSHL   (R3)                                            ;
                                69          04  FB 001A5     CALLS   #4, PAT$OUT_NUM_VAL                             ;
                                50          64  D0 001A8     MOVL    PAT$GB_MOD_PTR, R0                              ;4566
                                50      01  A0  9A 001AB     MOVZBL  1(R0), R0                                       ;
                                50          08  C4 001AF     MULL2   #8, R0                                          ;
            66      04  AC      50          00  EF 001B2     EXTZV   #0, R0, LOCATION, PAT$GL_LAST_VAL               ;
                                    30  AE  9F 001B8 12$:     PUSHAB  OUTPUT_BUFFER                                   ;4573
                    00000000G   EF      01  FB 001BB         CALLS   #1, PAT$OUT_PUT                                 ;
                                50          01  D0 001C2 13$:  MOVL    #1, R0                                      ;4575
                                            04 001C5         RET                                                     ;
                                        50  D4 001C6 14$:     CLRL    R0                                              ;4576
                                            04 001C8         RET                                                     ;

; Routine Size:  457 bytes,    Routine Base:  _PAT$CODE + 0D7A
```

```
;  1927          4577  1 ROUTINE DISPLAY_LVTS (VALUE) : NOVALUE =
;  1928          4578  1
;  1929          4579  1 !++
;  1930          4580  1 !
;  1931          4581  1 ! FUNCTIONAL DESCRIPTION:
;  1932          4582  1 !
;  1933          4583  1 !       Given a value, display the pathnames of all literals in the LVT which
;  1934          4584  1 !       have this value.
;  1935          4585  1 !
;  1936          4586  1 ! CALLING SEQUENCE:
;  1937          4587  1 !
;  1938          4588  1 !       CALLS #1, DISPLAY_LVTS
;  1939          4589  1 !
;  1940          4590  1 ! INPUTS:
;  1941          4591  1 !
;  1942          4592  1 !       VALUE - Literal value to be translated to symbols
;  1943          4593  1 !
;  1944          4594  1 ! IMPLICIT INPUTS:
;  1945          4595  1 !
;  1946          4596  1 !       The initial set up for standard PATCH I/O has already been done.
;  1947          4597  1 !       This routine (re)uses this buffer for its output.
;  1948          4598  1 !
;  1949          4599  1 ! OUTPUTS:
;  1950          4600  1 !
;  1951          4601  1 !       none
;  1952          4602  1 !
;  1953          4603  1 ! IMPLICIT OUTPUTS:
;  1954          4604  1 !
;  1955          4605  1 !       All the literal symbols associated with the value are printed.
;  1956          4606  1 !
;  1957          4607  1 ! ROUTINE VALUE:
;  1958          4608  1 !
;  1959          4609  1 !       novalue
;  1960          4610  1 !
;  1961          4611  1 ! SIDE EFFECTS:
;  1962          4612  1 !
;  1963          4613  1 !       Either output is sent to SYS$OUTPUT or a SIGNAL is generated and
;  1964          4614  1 !       no return is done.
;  1965          4615  1 !
;  1966          4616  1 !--
;  1967          4617  1
;  1968          4618  2 BEGIN
;  1969          4619  2
;  1970          4620  2 LOCAL
;  1971          4621  2         OUTPUT_BUFFER : REF VECTOR[,BYTE],                  ! Output buffer for SYS$OUTPUT writes
;  1972          4622  2         LVT_PTR : REF LVT_RECORD,                           ! Pointer to LVT match
;  1973          4623  2         ONE_FOUND;                                          ! Indicator if at least one symbol was found
;  1974          4624  2
;  1975          4625  2 !++
;  1976          4626  2 ! Initialize a flag which is used to know whether or not at least one match
;  1977          4627  2 ! to the given value has been found.  Also save a pointer to current output
;  1978          4628  2 ! buffer so that it can be reused.
;  1979          4629  2 !--
;  1980          4630  2 ONE_FOUND = FALSE;
;  1981          4631  2 OUTPUT_BUFFER = .PAT$CP_OUT_STR;
;  1982          4632  2
;  1983          4633  2 !++
```

```
;  1984          4634  2 ! Access to the LVT is via a 'canned' function.  Before using it, this routine
;  1985          4635  2 ! must signal its intention to do so.
;  1986          4636  2 !--
;  1987          4637  2 PAT$GET_NXT_LVT(SL_ACCE_INIT);
;  1988          4638  2
;  1989          4639  2 !++
;  1990          4640  2 ! Loop through the LVT sequentially, asking to see all currently valid records.
;  1991          4641  2 !--
;  1992          4642  3 WHILE ((LVT_PTR = PAT$GET_NXT_LVT(SL_ACCE_RECS)) NEQA 0)
;  1993          4643  3         DO
;  1994          4644  3         BEGIN
;  1995          4645  4         IF (.LVT_PTR[LVT_VALUE] EQL .VALUE)
;  1996          4646  3         THEN
;  1997          4647  4                 BEGIN
;  1998          4648  4                 LOCAL
;  1999          4649  4                         NT_PTR : REF NT_RECORD,
;  2000          4650  4                         PATH_VEC : PATHNAME_VECTOR;
;  2001          4651  4
;  2002          4652  4                 !++
;  2003          4653  4                 ! Found a match.  Print out the corresponding pathname by
;  2004          4654  4                 ! first building a pathname vector based on the returned NT_PTR.
;  2005          4655  4                 !--
;  2006          4656  4                 ONE_FOUND = TRUE;
;  2007          4657  4                 NT_PTR = .LVT_PTR[LVT_NT_PTR];
;  2008          4658  4                 PAT$ADD_NT_T_PV(.NT_PTR, PATH_VEC);
;  2009          4659  4                 PAT$PRINT_PATH(PATH_VEC);
;  2010          4660  4
;  2011          4661  4                 !++
;  2012          4662  4                 ! Write out the string and reset the global buffer pointers.
;  2013          4663  4                 !--
;  2014          4664  4                 PAT$OUT_PUT(.OUTPUT_BUFFER-1);
;  2015          4665  4                 PAT$CP_OUT_STR = .OUTPUT_BUFFER;
;  2016          4666  4                 PAT$GL_BUF_SIZ = 0;
;  2017          4667  3                 END;
;  2018          4668  2         END;                                          ! Loop back to consider the next LVT record
;  2019          4669  2
;  2020          4670  2 !++
;  2021          4671  2 ! At this point, the LVT has been completely searched.  If no matches were
;  2022          4672  2 ! found, then signal a warning.
;  2023          4673  2 !--
;  2024          4674  3 IF (NOT .ONE_FOUND)
;  2025          4675  2 THEN
;  2026          4676  2         SIGNAL(PAT$_NOLITERAL+MSG$K_WARN, 1, .VALUE);
;  2027          4677  2 RETURN;
;  2028          4678  1 END;                                                  ! End of DISPLAY_LVTS


                                        007C 00000 DISPLAY_LVTS:
                                                   .WORD   Save R2,R3,R4,R5,R6                     ; 4577
                    56 00000000G   EF   9E 00002           MOVAB   PAT$CP_OUT_STR, R6              ;
                    55 00000000G   EF   9E 00009           MOVAB   PAT$GET_NXT_LVT, R5             ;
                    5E             C2   C2 00010           SUBL2   #44, SP                         ;
                                   54   D4 00013           CLRL    ONE_FOUND                       ; 4630
                    53             66   D0 00015           MOVL    PAT$CP_OUT_STR, OUTPUT_BUFFER   ; 4631
```

```
                                 7E  D4 00018          CLRL    -(SP)                                   ; 4637
                              65 01  FB 0001A          CALLS   #1, PAT$GET_NXT_LVT
                                 01  DD 0001D 1$:      PUSHL   #1                                      ; 4642
                              65 01  FB 0001F          CALLS   #1, PAT$GET_NXT_LVT
                              52 50  D0 00022          MOVL    R0, LVT_PTR
                                 13  36 00025          BEQL    2$
                        04    AC 02  A2 D1 00027       CMPL    2(LVT_PTR), VALUE                       ; 4645
                                 12  EF 0002C          BNEQ    1$
                              54 01  D0 0002E          MOVL    #1, ONE_FOUND                           ; 4656
                              50 62  3C 00031          MOVZWL  (LVT_PTR), NT_PTR                       ; 4657
                            4001 8F  BB 00034          PUSHR   #^M<R0,SP>                              ; 4658
                00000000G   EF 02  FB 00038          CALLS   #2, PAT$ADD_NT_T_PV
                                 5E  DD 0003F          PUSHL   SP                                      ; 4659
                00000000G   EF 01  FB 00041          CALLS   #1, PAT$PRINT_PATH
                              FF A3  9F 00048          PUSHAB  -1(OUTPUT_BUFFER)                       ; 4664
                00000000G   EF 01  FB 0004B          CALLS   #1, PAT$OUT_PUT
                              66 53  D0 00052          MOVL    OUTPUT_BUFFER, PAT$CP_OUT_STR           ; 4665
                    00000000G EF  D4 00055           CLRL    PAT$GL_BUF_SIZ                          ; 4666
                                 C0  11 0005B          BRB     1$                                      ; 4642
                              12 54  E8 0005D 2$:      BLBS    ONE_FOUND, 3$                           ; 4674
                            04    AC  DD 00060         PUSHL   VALUE                                   ; 4676
                                 01  DD 00063          PUSHL   #1
                        006D82B8    8F  DD 00065       PUSHL   #7176888
                00000000G   00 03  FB 0006B          CALLS   #3, LIB$SIGNAL
                                 04  00072 3$:        RET                                             ; 4678

; Routine Size:  115 bytes,    Routine Base:  _PAT$CODE + 0F43
```

```
;2030     4679  1 GLOBAL ROUTINE PAT$REG_MATCH (STRING_DESC) =
;2031     4680  1
;2032     4681  1 !++
;2033     4682  1 !
;2034     4683  1 ! FUNCTIONAL DESCRIPTION:
;2035     4684  1 !
;2036     4685  1 !       Compares a string described by the string descriptor passed as the
;2037     4686  1 !       routine formal to each of the names of the machine registers.  If the
;2038     4687  1 !       string matches a register name, return the number of the register (0-16,
;2039     4688  1 !       where 16 is the PSL).  Otherwise return a -1.
;2040     4689  1 !
;2041     4690  1 ! CALLING SEQUENCE:
;2042     4691  1 !
;2043     4692  1 !       CALLS #1, PAT$REG_MATCH
;2044     4693  1 !
;2045     4694  1 ! INPUTS:
;2046     4695  1 !
;2047     4696  1 !       STRING_DESC - String descriptor to symbol string
;2048     4697  1 !
;2049     4698  1 ! IMPLICIT INPUTS:
;2050     4699  1 !
;2051     4700  1 !       The VAX machine register table.
;2052     4701  1 !
;2053     4702  1 ! OUTPUTS:
;2054     4703  1 !
;2055     4704  1 !       The number of the register whose name matched the string.
;2056     4705  1 !
;2057     4706  1 ! IMPLICIT OUTPUTS:
;2058     4707  1 !
;2059     4708  1 !       none
;2060     4709  1 !
;2061     4710  1 ! ROUTINE VALUE:
;2062     4711  1 !
;2063     4712  1 !       0-16 for the corresponding register
;2064     4713  1 !       -1 for no match
;2065     4714  1 !
;2066     4715  1 ! SIDE EFFECTS:
;2067     4716  1 !
;2068     4717  1 !       none
;2069     4718  1 !
;2070     4719  1 !--
;2071     4720  1
;2072     4721  2 BEGIN
;2073     4722  2
;2074     4723  2 MAP
;2075     4724  2         STRING_DESC : REF BLOCK [, BYTE];
;2076     4725  2
;2077     4726  2 LOCAL
;2078     4727  2         INDEX;
;2079     4728  2
;2080     4729  2 INDEX = 0;
;2081     4730  2 REPEAT
;2082     4731  3         BEGIN
;2083     4732  3         IF CH$EQL (.STRING_DESC [DSC$W_LENGTH], CH$PTR (.STRING_DESC [DSC$A_POINTER]),
;2084     4733  3                         .REGISTER_TABLE [.INDEX, REG_CH_CNT],
;2085     4734  3                         CH$PTR (REGISTER_TABLE [.INDEX, REG_NAME]))
;2086     4735  3         THEN RETURN .INDEX
```

```
;  2087          4736  3          ELSE
;  2088          4737  4                  BEGIN
;  2089          4738  4                  INDEX = .INDEX + 1;
;  2090          4739  4                  IF .INDEX GTR REGISTER_COUNT - 1
;  2091          4740  4                  THEN RETURN -1
;  2092          4741  3                  END;
;  2093          4742  2          END;
;  2094          4743  1 END;
;  INFO#212                 L1:4729
;  Null expression appears in value-required context
```

```
                                    003C 00000          .ENTRY  PAT$REG_MATCH, Save R2,R3,R4,R5          ; 4679
                                54  D4 00002          CLRL    INDEX                                    ; 4729
                            55      AC  D0 00004          MOVL    STRING_DESC, R5                          ; 4732
                  00000000'EF44     DF 00008 1$:        PUSHAL  REGISTER_TABLE[INDEX]                    ; 4733
                            50      9E  9A 0000F          MOVZBL  @(SP)+, R0
                  00000000'EF44     DF 00012          PUSHAL  REGISTER_TABLE+1[INDEX]                  ; 4734
        50          00      04  B5      04  BC  2D 00019          CMPC5   @STRING_DESC, @4(R5), #0, R0, @(SP)+
                                    9E  00020
                                04  12 00021          BNEQ    2$
                            50      54  D0 00023          MOVL    INDEX, R0                                ; 4735
                                    04 00026          RET
                                54  D6 00027 2$:       INCL    INDEX                                    ; 4738
                            10      54  D1 00029          CMPL    INDEX, #16                               ; 4739
                                DA  15 0002C          BLEQ    1$
                            50      01  CE 0002E          MNEGL   #1, R0                                   ; 4740
                                    04 00031          RET                                              ; 4743

; Routine Size:  50 bytes,    Routine Base:  _PAT$CODE + 0FB6
```

```
;  2096     4744  1 GLOBAL ROUTINE PAT$FILL_BUF(BUF_DESC, DATA_PTR, DATA_SIZ) : NOVALUE =
;  2097     4745  1
;  2098     4746  1 !++
;  2099     4747  1 !
;  2100     4748  1 ! FUNCTIONAL DESCRIPTION:
;  2101     4749  1 !
;  2102     4750  1 !       Takes the data defined as the input arguments and puts them in the
;  2103     4751  1 !       temporary deposit buffer.  This is accomplished by allocating a new
;  2104     4752  1 !       larger buffer, copying in the old buffer, and then deallocating it.
;  2105     4753  1 !       Then the buffer descriptor is updated.
;  2106     4754  1 !
;  2107     4755  1 ! CALLING SEQUENCE:
;  2108     4756  1 !
;  2109     4757  1 !       CALLS #2, PAT$FILL_BUF
;  2110     4758  1 !
;  2111     4759  1 ! INPUTS:
;  2112     4760  1 !
;  2113     4761  1 !       BUF_DESC - Buffer descriptor
;  2114     4762  1 !       DATA_PTR - Address of the data to be put in the buffer
;  2115     4763  1 !       DATA_SIZ - Number of bytes of data to be put in the buffer
;  2116     4764  1 !
;  2117     4765  1 ! IMPLICIT INPUTS:
;  2118     4766  1 !
;  2119     4767  1 !       none
;  2120     4768  1 !
;  2121     4769  1 ! OUTPUTS:
;  2122     4770  1 !
;  2123     4771  1 !       none
;  2124     4772  1 !
;  2125     4773  1 ! IMPLICIT OUTPUTS:
;  2126     4774  1 !
;  2127     4775  1 !       The buffer descriptor is updated.
;  2128     4776  1 !
;  2129     4777  1 ! ROUTINE VALUE:
;  2130     4778  1 !
;  2131     4779  1 !       none
;  2132     4780  1 !
;  2133     4781  1 ! SIDE EFFECTS:
;  2134     4782  1 !
;  2135     4783  1 !       The data is written into the buffer.
;  2136     4784  1 !
;  2137     4785  1 !--
;  2138     4786  1
;  2139     4787  2 BEGIN
;  2140     4788  2
;  2141     4789  2 MAP
;  2142     4790  2         BUF_DESC : REF BLOCK[,BYTE];                            ! Buffer descriptor
;  2143     4791  2
;  2144     4792  2 LOCAL
;  2145     4793  2         TEMP_PTR;                                               ! Pointer to new buffer
;  2146     4794  2
;  2147     4795  2 TEMP_PTR = PAT$FREEZ((.BUF_DESC[DSC$W_LENGTH] + .DATA_SIZ + A_LONGWORD -1)/A_LONGWORD); ! Allocate larger bu
;  2148     4796  2 IF .BUF_DESC[DSC$W_LENGTH] NEQ 0
;  2149     4797  2 THEN
;  2150     4798  3         BEGIN
;  2151     4799  3         CH$MOVE(.BUF_DESC[DSC$W_LENGTH], .BUF_DESC[DSC$A_POINTER], .TEMP_PTR); ! Move in previous data
;  2152     4800  3         PAT$FREERELEASE(.BUF_DESC[DSC$A_POINTER], (.BUF_DESC[DSC$W_LENGTH] +3)/4); ! Release old buffer
```

```
;  2153          4801  2          END;
;  2154          4802  2  CH$MOVE(.DATA_SIZ, .DATA_PTR, CH$PTR(.TEMP_PTR, .BUF_DESC[DSC$W_LENGTH])); ! Move in new data
;  2155          4803  2  BUF_DESC[DSC$A_POINTER] = CH$PTR(.TEMP_PTR);                                ! Reset buffer dsc addr
;  2156          4804  2  BUF_DESC[DSC$W_LENGTH] = .BUF_DESC[DSC$W_LENGTH] + .DATA_SIZ;   ! Reset buf dsc siz
;  2157          4805  1  END;


                                                  01FC 00000          .ENTRY  PAT$FILL_BUF, Save R2,R3,R4,R5,R6,R7,R8      ; 4744
                                        56    04  AC   D0 00002          MOVL    BUF_DESC, R6                                 ; 4795
                                        58        66   3C 00006          MOVZWL  (R6), R8                                     ;
                                50      58    0C  AC   C1 00009          ADDL3   DATA_SIZ, R8, R0                             ;
                                        50        03   C0 0000E          ADDL2   #3, R0                                       ;
                                7E      50        04   C7 00011          DIVL3   #4, R0, -(SP)                                ;
                                00000000G EF      01   FB 00015          CALLS   #1, PAT$FREEZ                                ;
                                        57        50   D0 0001C          MOVL    R0, TEMP_PTR                                 ;
                                                  58   D5 0001F          TSTL    R8                                           ; 4796
                                                  17   13 00021          BEQL    1$                                           ;
                                67      04  B6        58   28 00023          MOVC3   R8, @4(R6), (TEMP_PTR)                       ; 4799
                                        50        03  A8   9E 00028          MOVAB   3(R8), R0                                    ; 4800
                                7E      50        04   C7 0002C          DIVL3   #4, R0, -(SP)                                ;
                                        04    A6  DD   00030          PUSHL   4(R6)                                        ;
                                00000000G EF      02   FB 00033          CALLS   #2, PAT$FREERELEASE                          ;
                                6847    08  BC        0C  AC   28 0003A 1$:     MOVC3   DATA_SIZ, @DATA_PTR, (R8)[TEMP_PTR]  ; 4802
                                        04  A6        57   D0 00041          MOVL    TEMP_PTR, 4(R6)                              ; 4803
                                        66        0C  AC   A0 00045          ADDW2   DATA_SIZ, (R6)                               ; 4804
                                                      04   00049          RET                                                  ; 4805

; Routine Size:  74 bytes,    Routine Base:  _PAT$CODE + 0FE8
```

```
; 2159          4806  1 END
; 2160          4807  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _PAT$PLIT                      100  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;  _PAT$CODE                     4146  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                        0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619         7         0      1000          00:01.8


; Information:  1
; Warnings:     0
; Errors:       0


;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATEXA/OBJ=OBJ$:PATEXA MSRC$:PATEXA/UPDATE=(ENH$:PATEXA)

; Size:          4146 code + 100 data bytes
; Run Time:        01:20.5
; Elapsed Time:    04:10.7
; Lines/CPU Min:    3581
; Lexemes/CPU-Min: 28636
; Memory Used:  406 pages
; Compilation Complete
```

PATINT
LIS

PATINS
LIS

PATERR
LIS

PATIHD
LIS

PATEXA
LIS

PATFRE
LIS